UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
DEPARTAMENTO DE CIÊNCIAS AMBIENTAIS E TECNOLÓGICAS
CURSO DE BACHARELADO EM CIÊNCIA E TECNOLOGIA

APARECIDA BEZERRA DA SILVA

PROGRAMAÇÃO DE LEGO MINDSTORMS NXT 2.0 UTILIZANDO MATLAB®

MOSSORÓ
2013

APARECIDA BEZERRA DA SILVA

PROGRAMAÇÃO DE LEGO MINDSTORMS NXT 2.0 UTILIZANDO MATLAB®

Monografia apresentada à Universidade Federal Rural do Semi-Árido (UFERSA), Departamento de Ciências Ambientais e Tecnológicas para obtenção do título de Bacharel em Ciência e Tecnologia.

Orientador: Dr. Francisco José Targino Vidal – UFERSA.

MOSSORÓ
2013

Ficha catalográfica preparada pelo setor de classificação e catalogação da Biblioteca "Orlando Teixeira" da UFERSA

S586p Silva, Aparecida Bezerra da.

Programação de lego mindstorms NXT 2.0 utilizando matlab® / Aparecida Bezerra da Silva. – Mossoró, RN : 2013.
86f. : il.

Orientador: Profº. Dr. Francisco José Targino Vidal.
Monografia (Graduação) – Universidade Federal Rural do Semi-Árido, Graduação em Ciência e Tecnologia, 2013.

1. Robótica e educacional. 2. Lego mindstorms. 3. Matlab®.
I. Título.

CDD: 629.892

Bibliotecária: Marilene Santos de Araújo
CRB-5/1033

APARECIDA BEZERRA DA SILVA

PROGRAMAÇÃO DE LEGO MINDSTORMS NXT 2.0 UTILIZANDO MATLAB®

Monografia apresentada à Universidade Federal Rural do Semi-Árido (UFERSA), Departamento de Ciências Ambientais e Tecnológicas para obtenção do título de Bacharel em Ciência e Tecnologia.

Orientador: Dr. Francisco José Targino Vidal – UFERSA.

APROVADO EM 11 / 09 / 13

BANCA EXAMINADORA

Dr. Francisco José Targino Vidal
Orientador

Me. Amanda Gondim de Oliveira

Me. Francisco de Assis Brito Filho

Ao meu querido tio Manoel Bezerra (*In Memorian*) por sempre me fazer acreditar que iria “chegar longe”.

Aos meus pais Renan Bezerra e Antonia Marta Bezerra por todo amor, por sempre me apoiarem nos meus estudos e por fazerem tudo pela minha educação. Aos meus irmãos, Ramon Bezerra e Raiany Bezerra, à minha cunhada Niascara Caldas e à minha linda sobrinha Lara Sophia por sempre estarem presentes.

DEDICO

AGRADECIMENTOS

Agradeço Deus por Sua graça e amor. Por colocar pessoas maravilhosas em meu caminho. Pelo dom da vida que me é dado e por perdoar todas as minhas falhas.

Aos meus pais maravilhosos, Renan e Marta, por todo amor e dedicação. Por me ensinarem a respeitar a vida e agradecer por tudo que temos e somos. Pela educação e compromisso com o meu crescimento profissional. Por me amarem incondicionalmente.

Agradeço à minha vovó Antônia por todo amor e incentivo. Pelos almoços deliciosos aos domingos que me sempre fazem esquecer o mundo real.

Aos meus irmãos, Ramon e Raiany, por dividirem suas vidas comigo. À minha cunhada e à minha sobrinha Sophia por estarem presentes. E a minha prima Isabela Bezerra por me ajudar a esquecer da correria da faculdade e trabalho.

Ao meu orientador Francisco Vidal compreensão e ensino. Pela oportunidade de participar do projeto de inclusão digital usando robótica educacional que foi a base para o desenvolvimento deste trabalho.

Aos Professores Amanda Gondim e Francisco de Assis Brito pela disponibilidade em colaborarem com esse trabalho, fazendo parte da banca.

Aos meus queridos professores Valdenize Lopes, Fabrício Oliveira, Odacir Almeida, Marta Lígia, George Frederick e Thadeu Brandão pela diferença que fizeram na minha graduação.

Também agradeço aos professores da Engenharia Idalmir de Souza, Humberto Dionizio, Fabiana Varella, Adriano Aron, Marcílio Nunes e Sérgio Sombra por toda aluda durante as disciplinas eletivas.

Agradeço à minha segunda família, Francisca Lúcia, Sara Naftali e Emanuela Priscila por toda ajuda e compreensão durante anos e principalmente nesse período. Também à minha grande amiga Rhuana Lima por suportar minhas chatices e me ajudar bastante nesse período.

À amiga e colega de curso Enailma Luciana por me ajudar tanto, por me incentivar e também me dar broncas quando necessário. Pelas noites estudando juntas apesar de todo cansaço.

Aos meus amigos da graduação Mara Raquel, Ellen Correia, Lumena Cristine, Nataniel Wontoon, Hildemberg Nunes, Felipe Yuri, Edezio Dantas, George de Oliveira, Renan Abdon, Arimatéia Magno, Geraldo Reinaldo, Jhonnathas Ferreira, Fellipe Bastos, Solano Neto, Fernando Jales, Thomas Tadeu, Rafael Bezerra e Marcos Vinícius por toda ajuda, mesmo que não percebida por eles.

À Adriana Targino e Catherine Morgana que estão longe, mas me ajudaram em diversos momentos.

Agradeço a Hillman Ferreira pela ajuda nesse período e também pela compreensão com as minhas faltas no trabalho.

Agradeço também aos meus amigos e colegas da Orquestra Verbus por toda compreensão, na pessoa do Maestro Gleidson Rodrigues.

“Sem um fim Social o saber será a maior das futilidades”.

(Gilberto Freyre)

RESUMO

Silva, Aparecida Bezerra, **Programação de *LEGO MINDSTORMS NXT 2.0* utilizando *MATLAB*®.** Mossoró-RN, Universidade Federal Rural do Semi-Árido (UFERSA), Setembro de 2013.p.82. Trabalho de conclusão de curso bacharel em ciência e tecnologia. Orientador: Professor Dr. Francisco José Targino Vidal.

A programação de *LEGO MINDSTORMS NXT 2.0* utilizando a ferramenta *RWTH – Mindstorms Toolbox for MATLAB*® auxilia no ensino de lógica de programação aos ingressantes nos cursos de engenharia e tecnologia, sendo fundamentada no estado da arte que explora o princípio da robótica educacional e que proporciona ao aluno colocar em prática os conhecimentos adquiridos nos primeiros semestres dos cursos. O objetivo deste trabalho é desenvolver um conjunto de funções para a programação dos robôs, analisar os sinais dos dispositivos de entradas e saídas e ainda construir um passo a passo para implementação dos códigos a fim de obter um tutorial na língua portuguesa. Este trabalho foi desenvolvido com o auxilio dos *kits LEGO MINDSTORMS NXT 2.0*, do *software* computacional *MATLAB*® e uma ferramenta específica desenvolvida, para fins educacionais, pela *RWTH Aachen University* chamada *RWTH - Mindstorms Toolbox for MATLAB*®. Foi analisado cada resultado, gráfico gerado com as programações desenvolvidas para os sensores de cor e ultrassônico. Com este trabalho se conclui que a utilização da ferramenta é simples e de fácil compreensão e que pode complementar as disciplinas relacionadas à lógica, algoritmos e controle. Espera-se que cursos de introdução à robótica educacional sejam implantados na faculdade a fim de acompanhar o desempenho dos alunos ingressantes nos primeiros semestres dos cursos de engenharia e tecnologias, como também nos semestres mais avanços.

Palavras-Chave: Robótica Educacional, *LEGO MINDSTORMS NXT*, *MATLAB*®.

ABSTRACT

The programming of LEGO MINDSTORMS NXT 2.0 using the tool RWTH - Mindstorms Toolbox for MATLAB® assists in teaching logic programming to entering students in the courses of engineering and technology, being based on the state of the art that explores the principle of educational robotics and provides students to put into practice the knowledge acquired in the first semester of courses. The Objective of this work is to develop a set of functions for programming the robots, analyzing the signals from the devices of inputs and outputs and further build a walkthrough for implementation of codes in order to get a tutorial in Portuguese. This work was developed with the aid of LEGO MINDSTORMS NXT 2.0 kits, of MATLAB® computational software and a specific tool developed for educational purposes by RWTH Aachen University RWTH call - Mindstorms Toolbox for MATLAB®. Was analyzed each result and graph generated with the programming developed for color sensors and ultrasonic. With this work can be conclude that the utilization of the tool is simple and easy to understand and that can complement the disciplines related to Logic, Algorithms and Control. It is expected that courses of introduction to educational robotics being implanted in college in order to track the performance of students entering in the first semesters of courses in engineering and technology, but also in advances semesters.

Keywords: *Educational Robotics, LEGO MINDSTORMS NXT, MATLAB®.*

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

Robótica, por definição, é o conjunto de estudos e técnicas que visam substituir e auxiliar o homem nas suas funções motoras, sensoriais e intelectuais. O ensino da mesma tem abrangido o mundo inteiro sendo anexado nas mais diversas áreas e tem fundamental importância nas engenharias e áreas afins.

Existem muitas ferramentas desenvolvidas para o ensino da robótica, dentre as quais se destaca os *kits* da *LEGO MINDSTORMS NXT*, que podem ser utilizados para montagens de simples protótipos até ao ensino de conceitos mais avançados em engenharia como controle de sistemas.

A programação de robôs pode ser empregada como ferramenta que viabiliza o aumento da capacidade dos alunos na resolução de problemas, raciocínio lógico e trabalho em equipe. Neste trabalho, a programação de *LEGO MINDSTORMS NXT* dar-se-á através do *RWTH – Mindstorms NXT Toolbox for MATLAB*®, que foi desenvolvida pela *RWTH – Rheinisch-Westfälische Hochochule Aachen University* para fins educacionais, estabelecendo relações com as disciplinas de algoritmos e lógica de programação, sistemas de controle e análise de sinais e sistemas.

1.1. JUSTIFICATIVA DO TRABALHO

A programação de *LEGO MINDSTORMS NXT 2.0* utilizando *MATLAB*® justifica-se como ferramenta para ser desenvolvida no meio acadêmico por proporcionar aos alunos ingressantes nas áreas de tecnologias, desenvolvimento de sua propriedade intelectual além de incentivar a permanência nos cursos de engenharia e ciências exatas.

1.2. OBJETIVOS

Neste tópico serão abordados os objetivos, geral e específico, relacionados ao trabalho proposto.

1.2.1. Objetivo Geral

Desenvolver um conjunto de funções para programação de *LEGO MIDSTORMS NXT 2.0* utilizando *MATLAB*® como uma ferramenta auxiliar no ensino de lógica de programação oferecida aos ingressantes dos cursos de engenharia, assim como nas disciplinas em níveis mais avançados do curso.

1.2.2. Objetivos Específicos

- Programar robôs através do *MATLAB*®*;*
- Analisar os sinais dos dispositivos de *outputs* e *inputs*;
- Criar métodos e funções no *MATLAB*® para aplicação nos *kits LEGO MINDSTORMS NXT 2.0*;
- Desenvolver um tutorial para realização da programação.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. ROBÓTICA EDUCACIONAL

Há mais de 60 anos se iniciou a Era Digital com a criação do *ENIAC* que foi o primeiro computador digital. Desde então, a informática passou por grandes transformações ao ponto de termos tecnologias ultrapassadas em apenas alguns dias.

O ensino também passou grandes modificações com o início dessa era e nasceu o termo Robótica Educacional ou Pedagógica, que está fundamentada no trabalho de Seymour Papert e na sua teoria denominada *construcionismo* (BATISTA, 2010, p.16). Papert (1980, *apud* BATISTA, 2010) defende a abordagem *construcionista* e considera que os alunos aprendem melhor quando criam objetos ou artefatos externos que possam servir de apoio à construção interna do conhecimento. Ainda podem-se citar os pensamentos de Piaget (1972, *apud* SCHONS e*t al*), "as funções essenciais da inteligência consistem em compreender e inventar, em outras palavras, construir estruturas estruturando o real". Com isso se torna possível resolver os problemas propostos de forma dinâmica e lúdica, aumentando a capacidade de resolução de problemas.

A Robótica Educacional traz a vantagem de ser interdisciplinar e pode ser aplicada nas mais diversas áreas de ensino. Ela tem sido aplicada no colegial até aos cursos de ensino superior, como os de engenharia, por exemplo. De forma geral, para Simões *et al (*2003, *apud* GOMES *et al*, 2008), as principais vantagens pedagógicas da Robótica Educacional são:

- Transforma a aprendizagem em algo motivante, tornando bastante acessível os princípios de Ciência e Tecnologia aos alunos;
- Permite testar em um equipamento físico o que os estudantes aprenderam utilizando programas modelos que simulam o mundo real;
- Ajuda à superação de limitações de comunicação, fazendo com que o aluno verbalize seus conhecimentos e suas experiências e desenvolva sua capacidade de argumentar e conta-argumentar;
- Desenvolve o raciocínio e a lógica na construção de algoritmos e programas para controle de mecanismos;
- Favorece a interdisciplinaridade, promovendo a integração de conceitos de áreas como: matemática, física, eletrônica, mecânica e arquitetura.

Verifica-se a eficácia da mesma em vários projetos já implantadas nas escolas e faculdades pelo Brasil e exterior, uma delas é a *RWTH Aachen University*, na Alemanha, na qual Behrens *et al* (2010), afirma sobre a experiência com os calouros dos cursos de Engenharia Elétrica e Tecnologia da Informação:

> *"The percentage of students who passed the project, the unex-pectedly wide variety of robot systems created during the second project phase, and the evaluation results of both students and supervisors show that the project augmented student motivation and helped the students to develop their programming skills and acquire soft skills. The mathematical tasks, mechatronical issues, and inherent hardware limitations were challenging for most of the participants and led to intensive group dynamics, teamwork, competition, engagement, and remarkable creativity. Overall, the freshman introduction course met the practical engineering and MATLAB programming learning targets."*

2.2. *KIT LEGO MINDSTORMS NXT 2.0*

Os *kits LEGO MINDSTORMS* foram desenvolvidos com o objetivo de transformar o meio de ensino, permitindo aos alunos uma proximidade com o mundo real das tecnologias e engenharia. O primeiro lançamento deu-se em 2001 (SCHNEIDER, 2011, p.20), com um processador utilizado nas centrais *RCX* (*Robotic Commander Explorer*). Foi atualizado em 2006, baseado em um novo bloco central *NXT*, posteriormente foi atualizado para a versão *NXT* 2.0, lançado em 2009 com mais peças, mais variedades de sensores, *Bluetooth* e também com capacidade de calcular ponto flutuante. A última versão do *LEGO* chama-se *LEGO EDUCATION EV3*, mas no nosso trabalho será utilizada a versão do kit *LEGO MINDSTORMS NXT* 2.0. A Figura 1 mostra uma visão geral desta evolução.

Figura 1 – Evolução do *kit LEGO MINDSTORMS* a. *LEGO MINDSTORMS EDUCATION NXT BASE SET;* b. *LEGO MINDSTORMS NXT 2.0;* c. *LEGO EDUCATION EV3.*
Fonte: LEGO EDUCATION, 2013; LEGO MINDSTORMS, 2013; Lego do Brasil, 2013.

O *kit* possui um *smart brick*, que pode ser comparado a um *PLC - Programmable Logic Controller*, com suporte para três saídas (motores) e quatro entradas que são utilizadas para os mais diversos sensores. O *NXT* 2.0 possui três motores, quatro sensores, sendo um de cor, um ultrassônico, dois de toque e 619 peças para montagem dos protótipos. O *kit* também acompanha um *software* de fácil utilização que possui um sistema de "clica e arrasta" bastante intuitivo. A seguir serão descritas as características de cada componente.

2.2.1. *Smart Brick*

Segundo LEGO (2009), as especificações técnicas do *smart brick* são:

- *32-bit ARM7 microcontroller;*
- *256 Kbytes FLASH, 64 Kbytes RAM;*
- *8-bit AVR microcontroller;*
- *4 Kbytes FLASH, 512 Byte RAM;*
- *Bluetooth wireless communication (Bluetooth Class II V2.0 compliant);*
- *USB full speed port (12 Mbit/s);*
- *4 input ports, 6-wire cable digital platform (One port includes an IEC 61158 Type 4/EN 50 170 compliant expansion port for future use);*
- *3 outputs ports, 6-wire cable digital platform;*

- *100 x 64 pixel LCD graphical display;*
- *Loudspeaker – 8 kHz sound quality. Sound channel with 8-bit resolution and 2-16kHz sample rate;*
- *Power source: 6 AA batteries.*

A Figura 2 representa um *smart brick* do *LEGO MINDSTORMS NXT 2.0.*

Figura 2 – *Smart Brick*
Fonte: Autoria Própria

2.2.2. Motores

O *kit* possui três Servo Motores (Figura 3) interativos com sensor de rotação incluso em cada um. O sensor é capaz de medir a rotação (Uma rotação corresponde a 360 º) em graus positivos, sentido horário, e graus negativos, sentido anti-horário. É possível controlar seu funcionamento através das rotações e da velocidade. A Figura 4 mostra o motor internamente.

Figura 3 – Servo Motores

Fonte: Autoria Própria

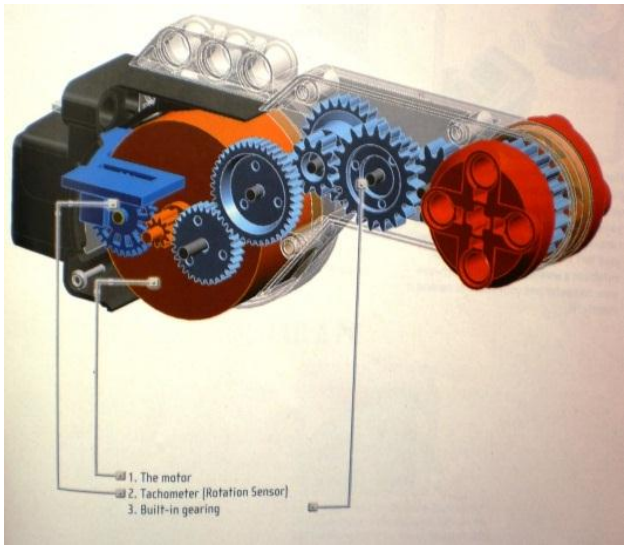

Figura 4 – Estrutura interna do Servo Motor

Fonte: *LEGO*, 2009.

2.2.3. Sensores de Toque

No *kit* há dois sensores de toque (Figura 5) que funcionam como um botão de pulso. Quando acionado permite a passagem do sinal elétrico, como mostra a Figura 6. Ele pode ser utilizado para executar ações quando for ativado (Sinal binário 1), desativado (Sinal binário 0), ou ainda quando houver um '*bumped*'.

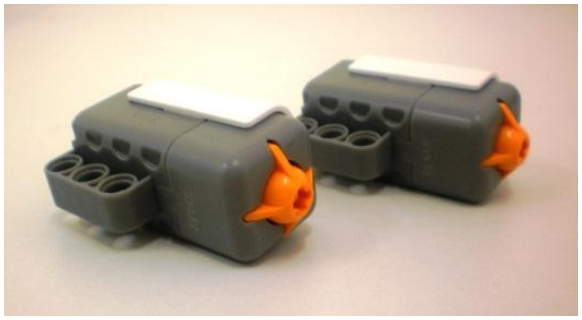

Figura 5 – Sensores de Toque

Fonte: Autoria Própria

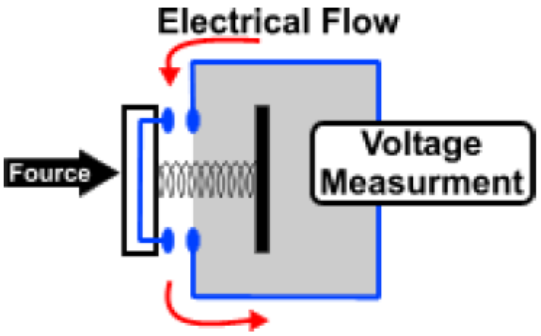

Figura 6 – Funcionamento do Sensor de Toque

Fonte: Batista, 2010.

2.2.4. Sensor de Cor

Há um sensor de cor presente no *kit* do tipo *RGB* e pode ser utilizado até de três formas diferentes. Ele pode distinguir a intensidade luminosa numa escala entre branco e preto; identifica até seis cores diferentes e a terceira função é a utilização como lâmpada.

Para utilizá-lo na identificação de cores (*Color Sensor*) é preciso aproximá-lo até 1 cm da superfície em ângulo reto. A função sensor de luz (*Light Sensor*) pode ser utilizada nas cores vermelha, verde e azul sendo indicada para ambientes com muita claridade ou superfícies refletoras e a função lâmpada (*Color Lamp*) é utilizada apenas para estética do protótipo. A Figura 7 apresenta um Sensor de Cor.

Figura 7 – Sensor de Cor

Fonte: Autoria Própria

2.2.5. Sensor Ultrassônico

O *kit* disponibiliza um sensor ultrassônico (Figura 8) que pode ser utilizado para identificar obstáculos, medir distâncias e detectar movimentos. A leitura pode ser feita em polegadas ou centímetros com um máximo de 255 centímetros e uma precisão de +/- 3 cm.

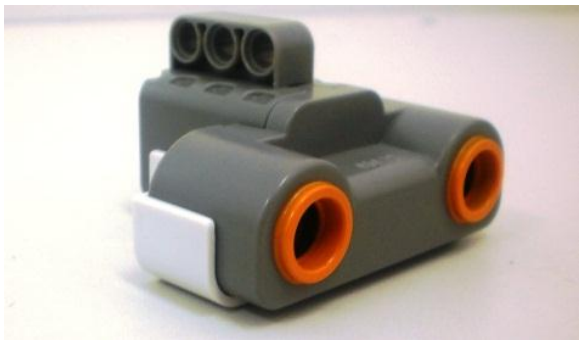

Figura 8 – Sensor Ultrassônico

Fonte: Autoria Própria

O seu princípio de funcionamento é baseado nos morcegos, onde uma onda sonora é enviada e a medida é feita pelo tempo de resposta entre a detecção do obstáculo e o retorno da onda como eco (Veja a Figura 9). Quanto maior a superfície melhor será a precisão do sensor, mas objetos com curvas, arredondados ou finos ele poderá não detectar.

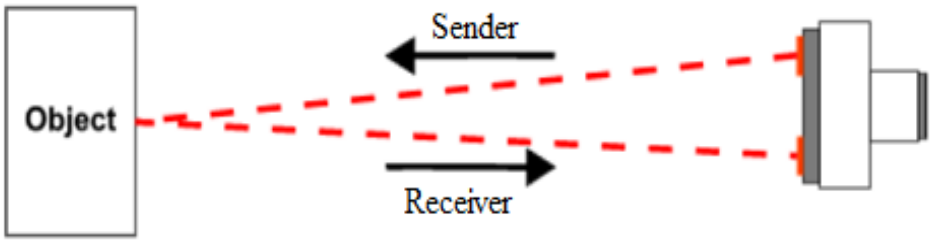

Figura 9 – Funcionamento do Sensor

Fonte: Batista, 2010.

2.3. *MATLAB*®

Segundo *Mathworks* (2012) o *MATLAB*® (*MATrix LABoratory*) é uma linguagem de alto nível e ambiente interativo para computação numérica, visualização e programação. Usando o *MATLAB*® é possível analisar dados, desenvolver algoritmos e criar modelos e aplicações. A linguagem, ferramentas e funções embutidas permitem explorar múltiplas abordagens e chegar a uma solução mais rápida do que comparado a linguagens de programa tradicionais como *C/C++* ou *Java*™. Há uma exibição do ambiente do *MATLAB*® na Figura 10.

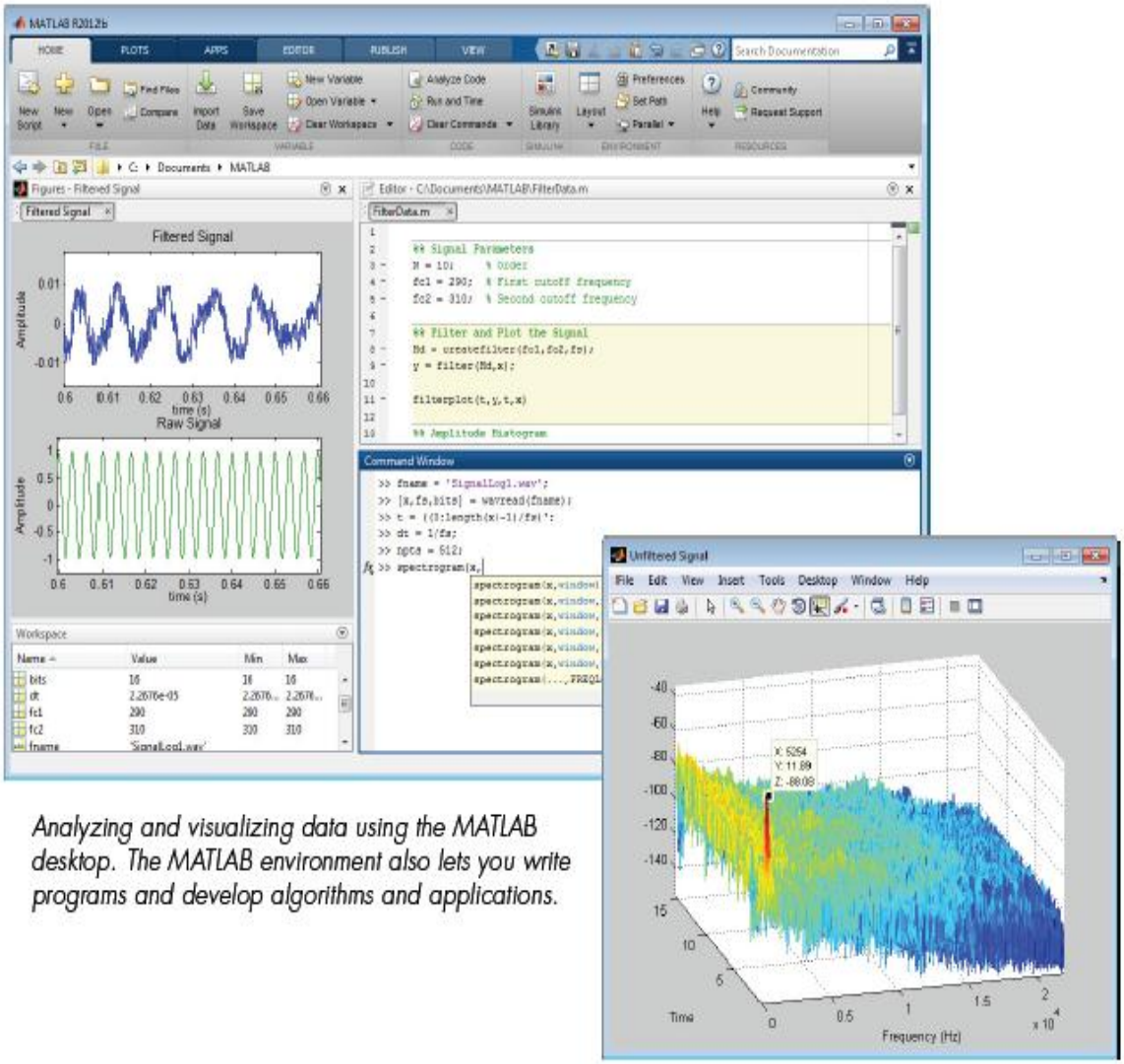

Figura 10 – *MATLAB*®

Fonte: *Mathworks*, 2012.

As características fundamentais do *MATLAB*® são abordadas abaixo (Adaptado de Mathworks, 2012):

- Linguagem de alto nível para com computação numérica, visualização e desenvolvimento de aplicações;
- Ambiente para exploração interativa, *design* e resolução de problemas;
- Funções matemáticas para álgebra linear, estatísticas análise de Fourier, filtragem, otimização, integração numérica e resolução de equações diferenciais ordinárias;
- *Built-in* gráficos para visualização de dados e ferramentas para a criação de gráficos personalizados;

- Ferramentas de desenvolvimento para melhorar a qualidade do código e facilidade de manutenção e maximização do desempenho;
- Ferramentas para construção de aplicações com interfaces gráficas personalizadas;
- Funções de integração *MATLAB*® baseados em algoritmos com aplicações externas e linguagens com *C, Java, NET* e *Microsoft Excel.*

2.3.1. *RWTH – Mindstorms NXT Toolbox for MATLAB*®

Entre as várias ferramentas do *MATLAB*® será trabalhada uma caixa de ferramentas (*Toolbox*) que foi desenvolvida pela *RWTH – Rheinisch-Westfälische Hochochule Aachen University*, o *RWTH – Mindstorms NXT Toolbox for MATLAB*®, que foi desenvolvido para controlar *LEGO MINDSTOMRS NXT* com o *MATLAB*® via *wireless Bluetooth* ou via *USB*. É um produto livre e está sujeito a *GNU General Public License (GLP)* e foi desenvolvido para estudantes de Engenharia Elétrica e projetado para fins educacionais.

O uso das funções para controle em tempo real não são muito recomendados via *wireless Bluetooth*, pois o tráfico de mensagens é lento e pode comprometer o projeto. Neste caso recomenda-se utilizar conexão via *USB* que possui um tempo de resposta mais rápido. De acordo com o *Institute of Imaging & Computer Vision* (2013), a principal vantagem da ferramenta é que podem ser combinadas aplicações para robôs com operações matemáticas complexas e suas visualizações no *MATLAB*®. Ela ainda permite ilimitadas possibilidades para fornecer aos robôs inteligência artificiais e outras melhorias usando os diversos recursos e cálculos do *MATLAB*® para processamento de sinais digitais.

Esta ferramenta foi utilizada com os alunos do primeiro semestre da *RWTH Aachen University* dos cursos de *Electrical Engineering, Information Technology* e *Technical Computer Science.* Os alunos aprenderam como usar o *MATLAB*® para processamento de sinal digital. Baseado na matemática básica os alunos mediram e analisaram as características dos diferentes atuadores e sensores do *LEGO Mindstorms NXT*. Depois, eles desenvolveram os seus próprios programas e robôs aplicando suas criatividades. No inverno de 2008 mais de 380 estudantes e 80 supervisores participaram deste projeto e desenvolveram 200 robôs utilizando *RWTH – Mindstorms NXT Toolbox for MATLAB*® (Adaptado de *Institute of Imaging & Computer Vision*, 2013). A Figura 11 mostra os trabalhos desenvolvidos pelos alunos.

Figura 11 – *The RWTH Aachen University first semester student laboratory - MATLAB® meets LEGO Mindstorms.*

Fonte: Adaptado de *Institute of Imaging & Computer Vision*, 2013.

2.4. OUTROS *SOFTWARES* PARA PROGRAMAÇÃO DE *LEGO MINDSTORMS NXT 2.0*

Os kits *LEGO MINDSTORMS NXT 2.0* podem ser programados com outras ferramentas, entre as quais o *software* da *LEGO MINDSTORMS NXT®*, RoboEduc – Robótica Pedagógica e *Bricx Command Center*.

2.4.1. *LEGO MINDSTORMS NXT®*

Este *software* foi desenvolvido e distribuído pelo *LEGO Group* para programação dos *kits LEGO*. Ele é indicado para crianças a partir de 10 anos e possui uma interface gráfica bastante intuitiva com um sistema de "clica e arrasta" para programação em blocos, permitindo programar robôs simples e complexos. A Figura 12 apresenta uma ilustração da interface gráfica do *software*.

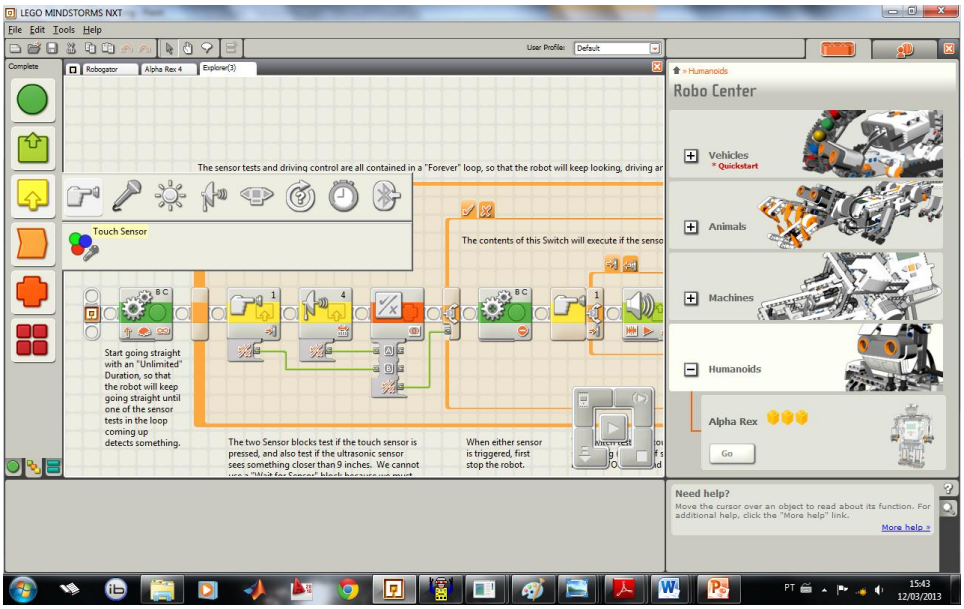

Figura 12 – Interface Gráfica do *Software LEGO MINDSTORMS NXT®*

Fonte: Autoria Própria

Este *software* possui ainda quatro protótipos com o passo a passo disponível para desenvolvimento da montagem e da programação (Figuras 13 e 14), além de um tutorial em forma de vídeo (Figura 15) com as principais funções do mesmo.

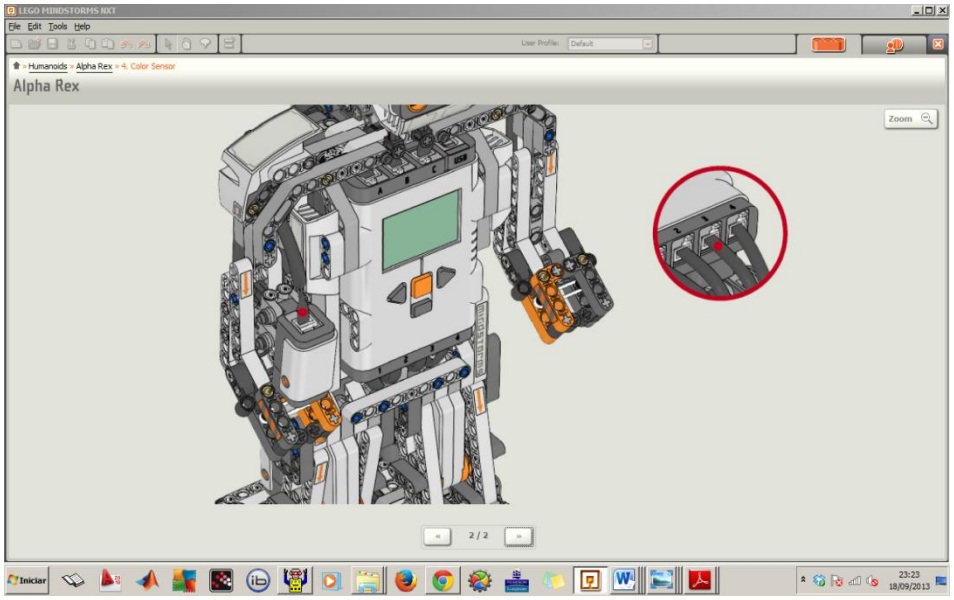

Figura 13 – Montagem dos Robôs (*Software LEGO MINDSTORMS NXT*®)

Fonte: Autoria Própria

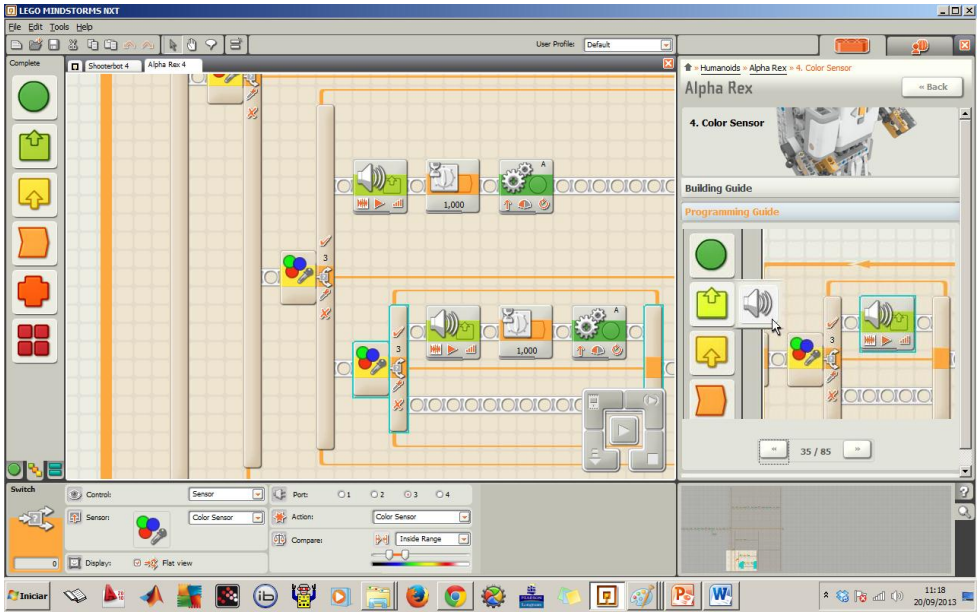

Figura 14 – Guia de Programação (*Software LEGO MINDSTORMS NXT*®)

Fonte: Autoria Própria

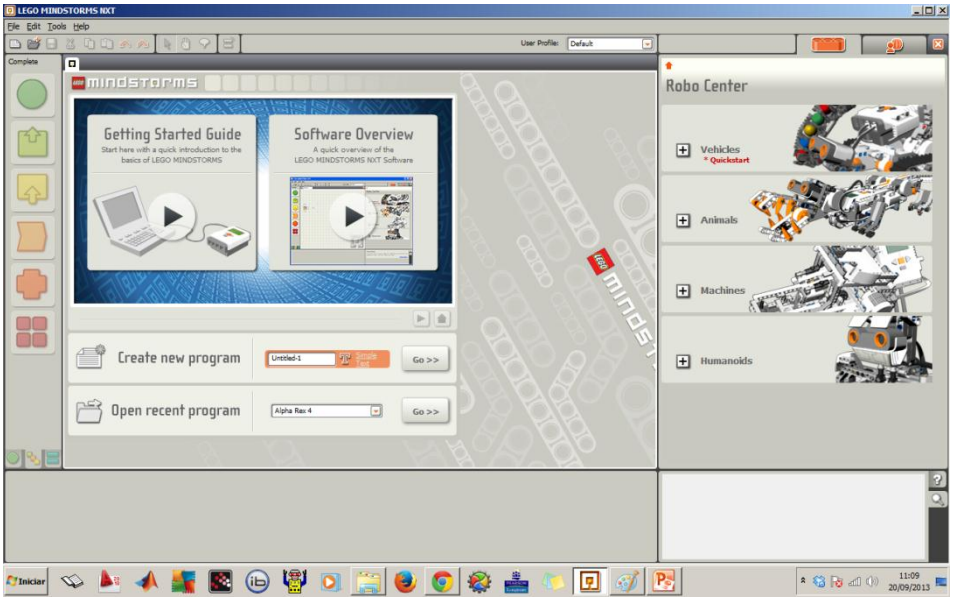

Figura 15 – Tutorial em forma de vídeos (*Software LEGO MINDSTORMS NXT*®)

Fonte: Autoria Própria

2.4.2. RoboEduc – Robótica Pedagógica

O RoboEduc – Robótica Pedagógica foi desenvolvido com o intuito de proporcionar a inclusão digital para alunos iniciantes na vida escolar (BARROS, 2006). É uma ferramenta simples e bastante intuitiva que possui cinco níveis de linguagens de programação. Pode ser caracterizada com uma ferramenta multi-plataforma, que pode ser executado em mais de um sistema operacional, tratando-se de um *software* livre e com sua licença de uso gratuita. Ele possui dois ambientes (Figuras 16 e 17), uma para o aluno e outro para o professor planejar e projetos os protótipos.

Há a opção do nível de programação que varia de 1 a 5 (Figura 18), respectivamente da linguagem de alto nível para baixo nível, ou seja, os níveis de 1 a 4 possuem programação com blocos e o nível 5 utiliza a programação totalmente em linhas de código com uma linguagem próxima a "*C*", como mostra a Figura 19.

Figura 16 – Interface Gráfica do RoboEduc

Fonte: Autoria Própria

Figura 17 – Modelos dos Protótipos do RoboEduc

Fonte: Autoria Própria

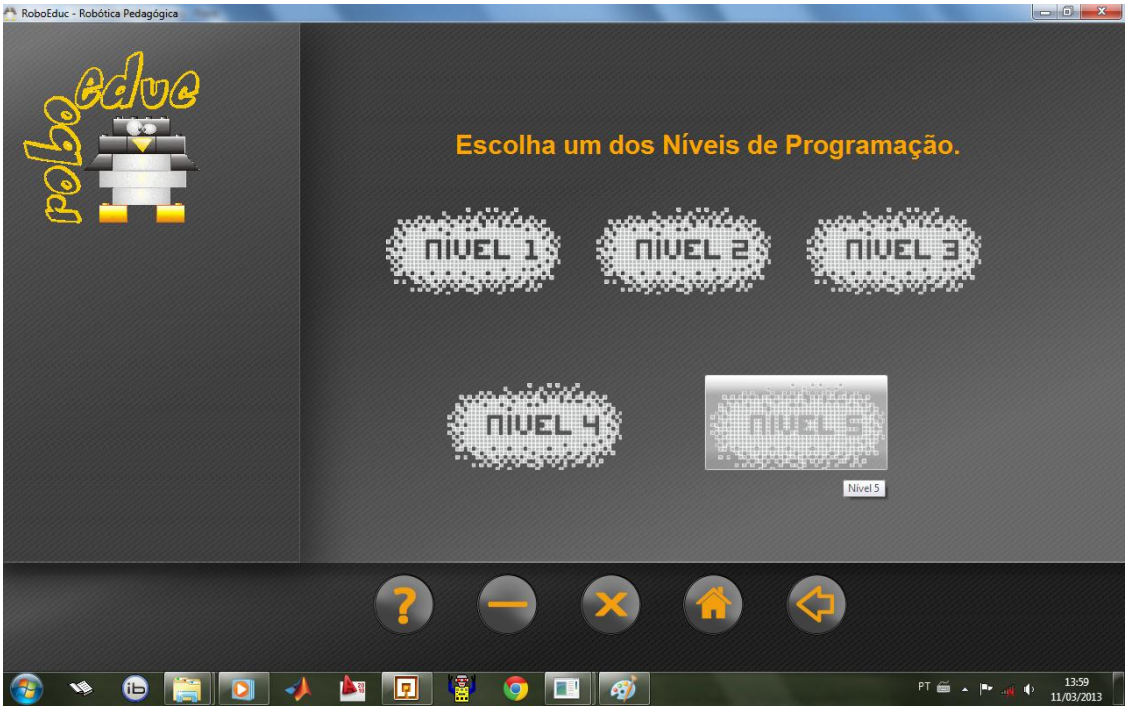

Figura 18 – Níveis de Programação (RoboEduc)

Fonte: Autoria Própria

Figura 19 – Linguagem de Programação Nível 5 (RoboEduc)

Fonte: Autoria Própria

2.4.3. *Bricx Command Center*

O *Bricx Command Center (BricxCC)* é um programa de 32 bits do *Windows* comumente conhecido como um ambiente de desenvolvimento integrado (IDE) para programação de robôs *LEGO MINDSTORMS* de todas as gerações da família *MINDSTORMS*, incluindo a terceira geração *EV3*. O *BricxCC* é totalmente compatível com todas as versões do *Windows*, incluindo 64-bit do *Windows* 7 e 8, juntamente com as versões de sistemas operacionais de servidor *Windows* (Adaptado de SOURCE FORGE, 2013).

O *BricxCC* suporta a programação *RCX* (todas as versões), *Scout*, *Cybermaster* e *Spybot* programáveis usando *Not Quite C* (*NQC*) linguagem de Dave Baum. E suporta a programação do *Scout*, *RCX2* e *Spybot* usando *Mindscript* do Grupo *LEGO (tm)* e *LASM* linguagens (*tm*) via *SDK* do *MINDSTORMS 2.5*. Ele também suporta programação *RCX bricks* em *Forth*, *C*, *C ++*, *Pascal* e *Java* usando *cygwin* juntamente com *pbForth*, *BrickOS* e *leJOS firmwares* alternativos (Adaptado de SOURCE FORGE, 2013). A Figura 20 mostra o ambiente de programação do *BrickxCC.*

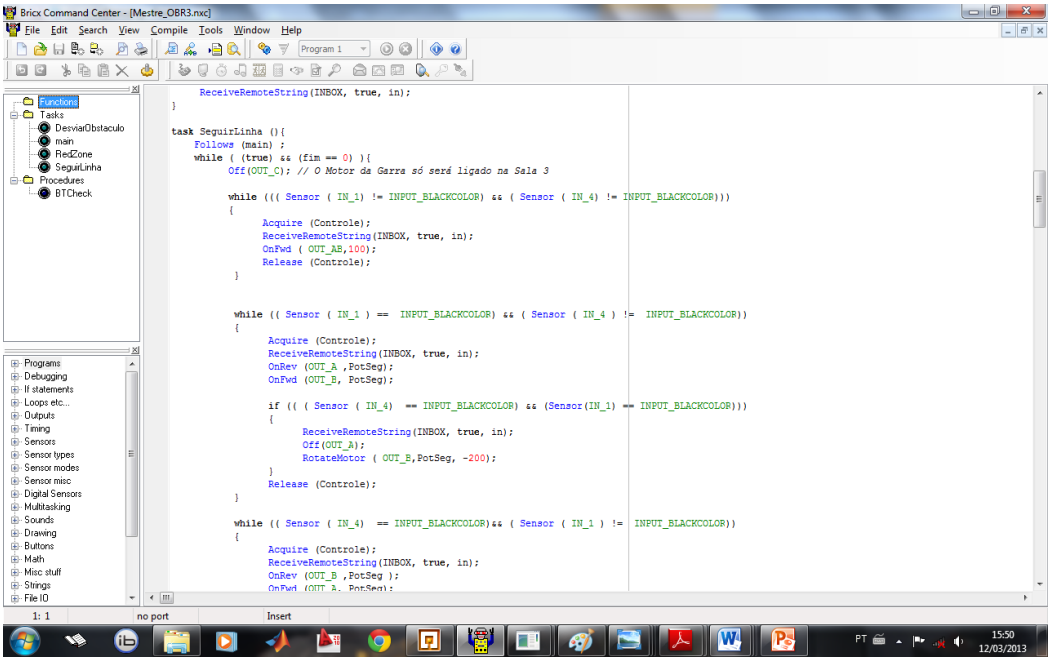

Figura 20 – Ambiente de Programação do *BricxCC*

Fonte: Autoria Própria

3. MATERAIS E PROCEDIMENTOS

O desenvolvimento de funções para a programação de *LEGO MINDSTORMS NXT 2.0*, utilizando *MATLAB*®, será fundamentada no estado da arte, que explora o uso da robótica na criação de ambientes de aprendizagem nos primeiros anos dos cursos de engenharia, para realização dos primeiros experimentos. A análise dos experimentos será realizada em conjunto com a implementação das funções a partir da execução de robôs, verificando a eficácia dos procedimentos para aplicação dos mesmos às disciplinas dos cursos de engenharia e tecnologias.

Este trabalho de conclusão de curso foi baseado nos artigos: *MATLAB*® *Meets LEGO Mindstorms – A Freshman Introduction Course into Practical Engineering* (BEHRENS *et al*, 2010) e *Signal Processing Experiments with the LEGO MINDSTORMS NXT Kit for Use in Signals and Systems Courses* (FERRI *et al*, 2009).

A execução das etapas que foram desenvolvidas apresenta-se no fluxograma da Figura 21.

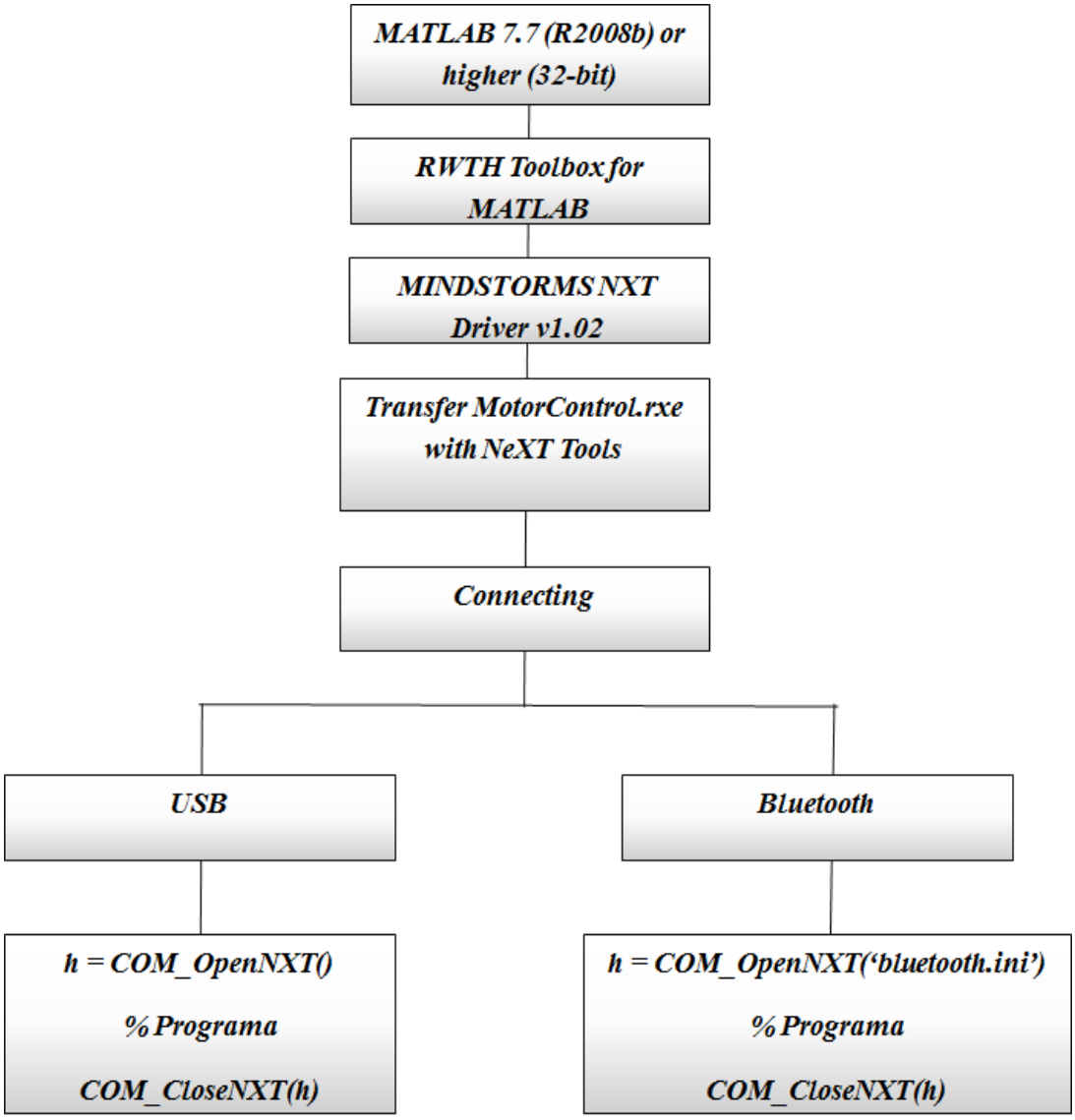

Figura 21 – Fluxograma Geral da metodologia que foi utilizada para Programação de *LEGO MINDSTORMS NXT 2.0* utilizando *MATLAB*®

Fonte: Elaboração Própria

O trabalho foi realizado no Laboratório de Robótica do Departamento de Ciências Ambientais e Tecnológicas (DCAT) da Universidade Federal Rural do Semi-Árido (UFERSA), Mossoró – RN. O *kit LEGO MINDSTORMS NXT* 2.0 utilizado para o desenvolvimento do trabalho pertence à instituição.

Para iniciar a programação do *LEGO* é necessário seguir os seguintes passos:

1. Instalar o *MATLAB*®, a partir da versão 7.7, R2008b (32-bit);
2. Instalar a ferramenta *RWTH Toolbox for MATLAB*®;
3. Instalar o drive de comunicação *MINDSTORMS NXT Driver v1. 02* para utilização da porta *USB*;

4. Transferir o arquivo *MotorControl.rxe* para o *smart brick*;
5. Conectar o *LEGO* com o computador via *USB* ou *Bluetooth*;
6. E, no *MATLAB®*, inserir os códigos necessários para controle do protótipo.

3.1. *RWTH TOOLBOX FOR MATLAB®*

Com o *MATLAB®* instalado, o segundo passo é adicionar o *toolbox* da ferramenta obedecendo a seguinte sequência: *(1) Home; (2) Set Path; (3) Add Folder: RWTHMindstormsNXT; RWTHMindstormsNXT / tools* e *RWTHMindstormsNXT / demos* e *(4) Save.*

3.2. *MOTOR CONTROL*

Motor Control é um programa que vem com a caixa de ferramentas *RWTH – Mindstorms NXT* e que deve ser transferido para o *smart brick*. Ele é compilado do *NXC/NBC* para um arquivo do tipo *RXE*. Ele permanece funcionando no *NXT* e recebendo os comandos que estão sendo executados no *MATLAB®*. Abaixo estão listadas as características do *Motor Control* (*INSTITUTE OF IMAGING & COMPUTER VISION*, 2013).

- Movimento preciso dos motores para qualquer posição (Precisão de ± 1grau de precisão em 90% do tempo);
- Parada suave;
- *Anti-stall-protection* (O motor não fica preso durante a desaceleração)*;*
- Aceleração suave opcional*;*
- Suporte para condução sincronizada para as duas rodas;
- Controla os três motores de forma independente;
- Movimentos / comandos podem ser cancelados a qualquer momento;
- Motores e sensores podem ser monitorados todo o tempo;
- Fim do movimento pode ser detectado.

3.2.1. Instalação do *MotorControl.rxe*

Após a execução dos passos 1, 2 e 3 descritos anteriormente, é preciso transferir o arquivo *MotorControl.rxe* para o *smart brick* utilizando o executável *NeXTExplorer*, na seguinte ordem:

1. Conectar o *NXT* via *USB* e inicia-lo;
2. Executar o *NeXTExplorer.exe* e selecionar a porta *USB;*
3. Transferir o arquivo *MotorControl22.rxe*, ou uma versão mais atual, para o *smart brick* como mostra a Figura 22. O arquivo se encontra na pasta *RWTHMindstorms / Tools / MotorControl*.

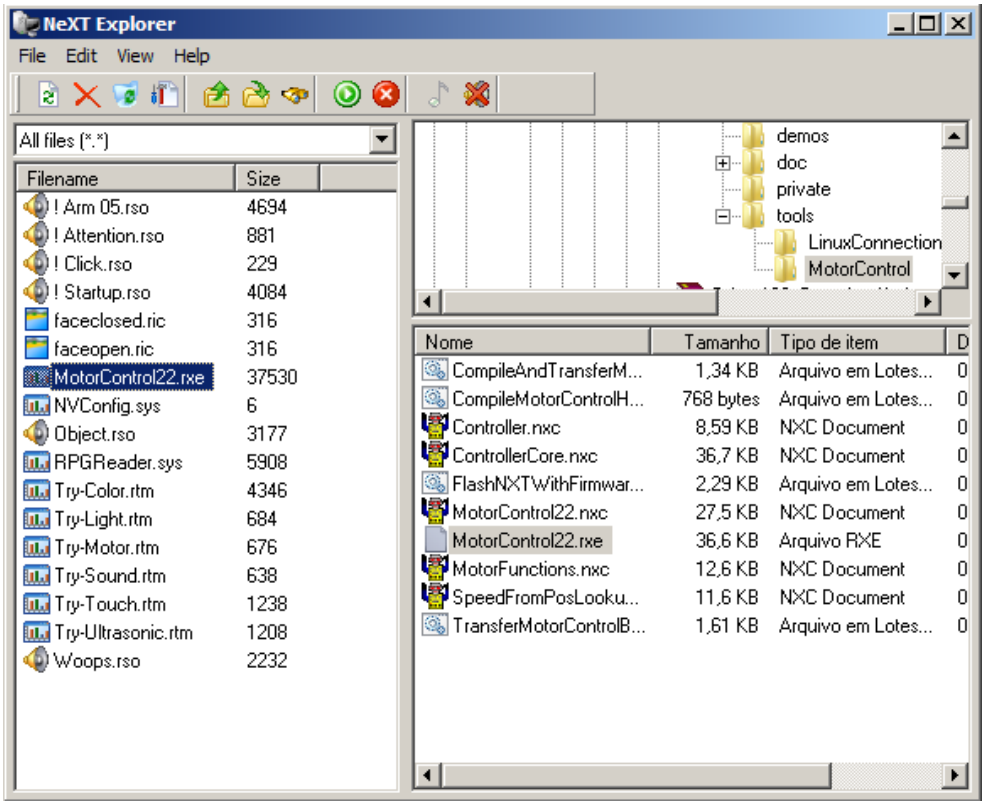

Figura 22 – Instalação do arquivo *Motor Control*

Fonte: Autoria Própria

4. Feche o *NeXTExplorer.exe.*

3.3. CONEXÃO

A conexão do *MATLAB*® com o *NXT* pode ser feita via *USB* / *Bluetooth*. Neste trabalho será utilizada apenas a comunicação via *USB*, mas será mostrado o passo a passo para conexão via *Bluetooth.*

3.3.1. Conexão via *USB*

Ao conectar o cabo *USB* e iniciar o *NXT*, deve-se iniciar a comunicação através dos seguintes códigos presentes na ferramenta *RWTH – Mindstorms NXT Toolbox for MATLAB*®*:*

```
h = COM_OpenNXT();
COM_SetDefaultNXT( h )
NXT_PlayTone( 440, 100 )

%Code

COM_CloseNXT(h);
```

A função *COM_OpenNXT* retorna uma "*handle*", uma espécie de referência criada na conexão via *USB* ou *Bluetooth.* As funções do *RWTH Toolbox* necessitam das informações contidas nessa *handle* para enviarem os dados dos sensores ou as configurações dos motores (Adaptado de *CLEMSON*, 2013).

3.3.2. Conexão via *Bluetooth*

O processo para comunicação via *Bluetooth* é semelhante ao anterior. A diferença básica é que primeiro deve conectar o *NXT* via *Bluetooth* seguindo o passo a passo a seguir:

1. Ir a Painel de Controle; *Hardware* e sons; Dispositivos e Impressoras; Adicionar um dispositivo;
2. Selecionar o *NXT* e entrar com o código "1234";
   Depois que o dispositivo for adicionado, verificar em propriedades de *hardware* em qual porta de comunicação ele está conectado;
3. Acessar o *MATLAB*® e digitar o código: COM_MakeBTConfigFile e clique em "*yes"* para confirmar;
4. Configurar a "*SerialPort*" em que o *NXT* está conectado (Figura 23) e clicar em "*Ok*".

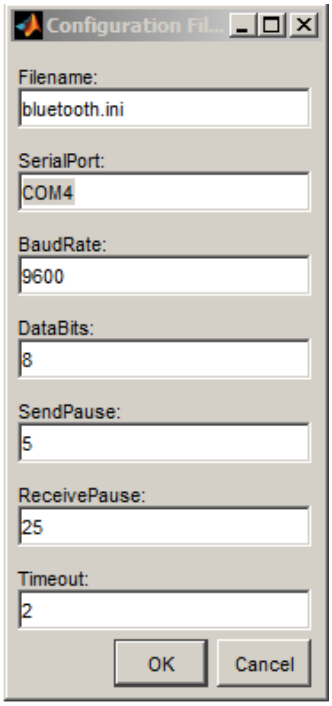

Figura 23 – Configuração da Porta Serial

Fonte: Autoria Própria

5. Depois acessar o *MATLAB*® entrar com o seguinte código:

```
h = COM_OpenNXT('bluetooth.ini');
COM_SetDefaultNXT( h )
NXT_PlayTone( 440, 100 )

    %Code

COM_CloseNXT(h);
```

3.4. CONFIGURAÇÃO DOS DISPOSITIVOS *"OUTPUTS"* E *"INPUTS"*

Para os motores e sensores há funções especificas para escolher seus parâmetros e configurações.

3.4.1. Configuração dos Parâmetros dos Motores

Podem ser configurados os valores de potência, direção e rotação dos motores. O primeiro passo é definir uma variável para herdar os atributos do motor. Para isso utilizamos o código abaixo, definindo a entrada do motor conectado no *smart brick* que se deseja controlar:

```
m = NXTMotor( MOTOR_A )
```

Ao executar este comando no *MATLAB*®, visualizam-se os seguintes parâmetros:

```
NXTMotor object properties:

            Port(s): 0           (A)
              Power: 0
    SpeedRegulation: 1           (on)
        SmoothStart: 0           (off)
         TachoLimit: 0           (no limit)
 ActionAtTachoLimit: 'Brake'     (brake, turn off when stopped)
```

Para modificar a potência do motor é preciso incrementar o valor em porcentagem, entre -100 e 100 %. O valor negativo corresponde ao giro no sentido anti-horário. A seguir há alguns exemplos:

```
m.Power = 50                    % Determina a potência do motor
m.SpeedRegulation = true        % Regula a velocidade alcançando %rotação
constante
m.SmoothStart = true            % Aceleração Suave
m.TachoLimit = 360              % Graus que o motor vai girar antes de
%parar
m.ActionAtTachoLimit = 'Coast' % Ação do motor após parar %('Coast','Brake'
e 'Holdbrake)
m.SendToNXT()                   % Envia as configurações e as executa
m.Stop('off')                   % Para o motor com a função 'Coast'
m.Stop('brake')                 % Para o motor com freio
```

Para fazer o motor andar pode-se determinar a mesma variável para controle dos dois motores:

```
mBC = NXTMotor( [ MOTOR_B; MOTOR_C ] )
mBC.Power = 50
mBC.SendToNXT()
```

Pode-se ainda utilizar o código: `DirectMotorCommand( port, power, tachometerLimit, speedRegulation,syncToMotor, turnRatio, smoothStart )`, que é semelhante ao anterior, veja o exemplo:

```
DirectMotorCommand( MOTOR_B, 50, 0, 'off', MOTOR_C, 0, 'off' )
mBC.SendToNXT()
```

3.4.2. Configuração dos Parâmetros do Sensor de Toque

O sensor de toque retorna os valores binários (0 ou 1) quando utilizado no *MATLAB*®. A sequência de códigos é muito simples:

```
OpenSwitch( SENSOR_1 ) % Configura porta 1 para o sensor de toque
GetSwitch( SENSOR_1 )  % retorna o valor do sensor, 0 ou 1
CloseSensor( SENSOR_1 )% Fecha a comunicação com o sensor
```

A porta utilizada pelo sensor pode ser especificada de 1 a 4 de acordo com a numeração do *smart brick.*

3.4.3. Configuração dos Parâmetros do Sensor de Cor

O sensor de cor pode ser utilizado para identificar as cores através de duas funções: *Color Sensor* e *Ligth Sensor*. Na função cor, o sensor consegue distinguir as cores preta, branca, vermelha, amarela, verde e azul. A resposta é o próprio nome da cor. Já na função luz, o sensor retorna um valor numérico correspondente de 0-1023 de acordo com cada superfície e quando mais brilhante a cor, maior será o valor numérico de retorno.

Para a função cor, utiliza-se a seguinte sequência de códigos:

```
OpenNXT2Color( SENSOR_1, 'FULL' ) % Escolhe a função cor
GetNXT2Color( SENSOR_1 )          % Retorna o nome da cor visualizada
CloseSensor( SENSOR_1 )           % Encerra a comunicação com o sensor
```

Para a função luz, se pode escolher a cor do sinal, que pode ser vermelha, azul ou verde:

```
OpenNXT2Color( SENSOR_1, 'RED' ) % BLUE ou GREEN
GetNXT2Color( SENSOR_1 )         % Retorna valores de 0-1023
CloseSensor( SENSOR_1 )          % Encerra a comunicação com o sensor
```

3.4.4. Configuração dos Parâmetros do Sensor Ultrassônico

O sensor ultrassônico pode ser utilizado em dois modos: o *continuous mode* e *snapshot mode*. No *continuos mode* o sensor ultrassônico envia o sinal sonoro continuamente fazendo a medição dos ecos. No *snapshot mode* o sensor só envia o sinal apenas uma vez quando solicitado, isso evita conflitos entre os sinais quando existirem mais sensores ultrassônicos na mesma área.

Para utilizar o sensor no *continuos mode* utilize o código a seguir:

```
OpenUltrasonic( SENSOR_1 ) % Ativa o sensor
pause( 0.1 )               % Tempo para o sensor medir o eco
GetUltrasonic( SENSOR_1 )  % Obtendo leitura
CloseSensor( SENSOR_1 )    % Desligar conexão com o sensor
```

E para utilizar o sensor no *snapshot mode* utilize o código a seguir:

```
OpenUltrasonic( SENSOR_1, 'SNAPSHOT' )
USMakeSnapshot( SENSOR_1 )
pause( 0.1 )
eco = USGetSnapshotResults( SENSOR_1 )
eco(1)
```

3.5. PROGRAMAS DESENVOLVIDOS

Para cada tipo sensor foi desenvolvido um ou mais programas que serão mostrados nos tópicos seguintes. O protótipo montado para execução de todos os programas se chama *Five Minute Bot* (NXT PROGRAMS, 2011). Algumas adaptações foram realizadas para melhor adequação, como mostra a Figura 24.

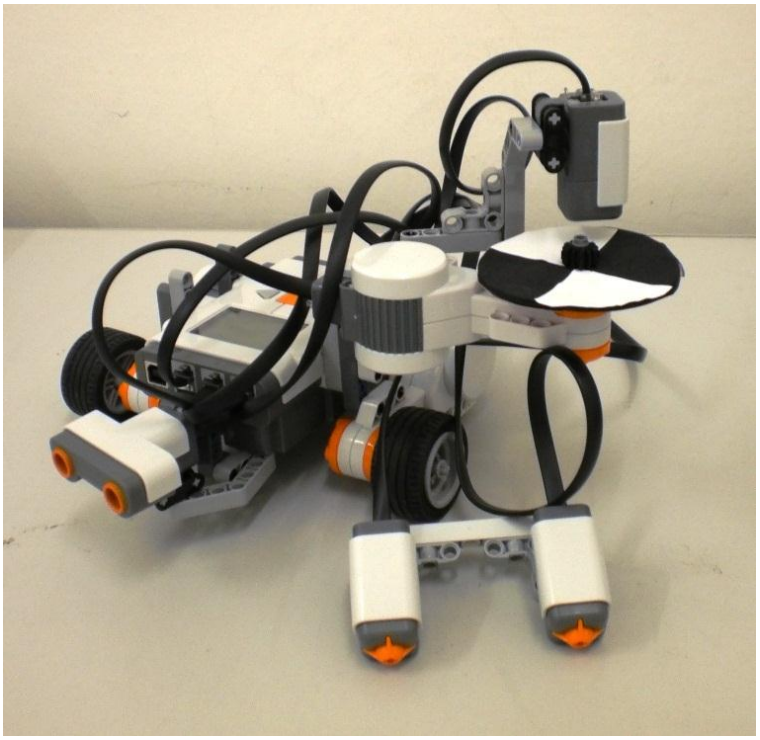

Figura 24 – Protótipo utilizado

Fonte: Elaboração Própria (Adaptado de NXT PROGRAMS, 2011).

3.5.1. Programa 1 – Controle Simples Utilizando Uma Interface Gráfica

Esta interface gráfica (Figura 25) foi criada para controlar o protótipo com o *mouse*. Ela controla os três motores e permite que robô gire para direita e esquerda, siga em frente ou de ré e aciona o motor “A” no sentido horário e anti-horário.

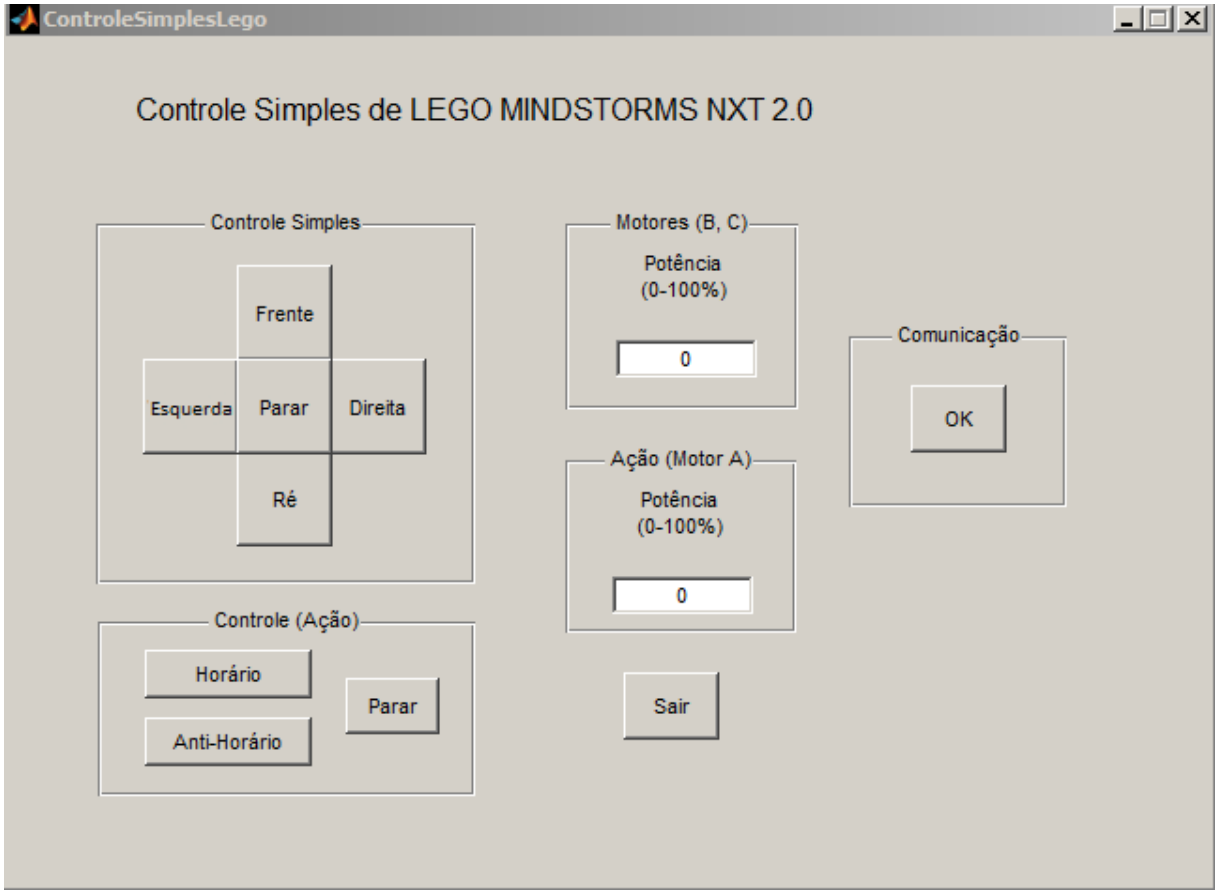

Figura 25 – Programa 1 (Interface Gráfica)

Fonte: Elaboração Própria

A programação completa do Programa 1 pode ser visualizada no Anexo A.

3.5.2. Programa 2 – *Touch Control*

O Programa 2 foi desenvolvido para controlar o protótipo proposto utilizando dois sensores de toque como controle, como pode ser visto na Figura 26. Ao acionar o "botão" da esquerda, o robô deve girar para esquerda; ao acionar o "botão" da direita, deve girar para a direita e, quando os dois são acionados ao mesmo tempo, o robô segue em frente. Veja o código utilizado a seguir:

```
%%Check toolbox installation: verify that the RWTH - Mindstorms NXT
%toolbox is installed.
if verLessThan('RWTHMindstormsNXT', '2.00');
    error(strcat('This program requires the RWTH - Mindstorms NXT Toolbox '
...
        ,'version 3.00 or greater. Go to http://www.mindstorms.rwth-
aachen.de ' ...
        ,'and follow the installation instructions!'));
```

```
end%if
%% Clean up: Close previous handles (if existing)
close all
clear all
clc
format compact
%% Iniciando comunicação com o LEGO
h = COM_OpenNXT();
COM_SetDefaultNXT( h )
NXT_PlayTone( 440, 100 )

%% Ativando o Sensores de Toque e Motores
me = NXTMotor( MOTOR_B ); % Motor da Esquerda
md = NXTMotor( MOTOR_C ); % Motor da Direita

OpenSwitch(SENSOR_1)
OpenSwitch(SENSOR_2)

%% Função Controle
while(true)
    if GetSwitch (SENSOR_1)
        if GetSwitch (SENSOR_2)
            %Frente
            me.Power = 25;
            md.Power = 25;
            me.SendToNXT()
            md.SendToNXT()
        else
            %me.Stop('Off')
            %Esquerda
            md.Power = 25;
            me.Power = -25;
            me.SendToNXT()
            md.SendToNXT()
        end
    else
        if GetSwitch (SENSOR_2)
            %md.Stop('Off')
            %Direita
            me.Power = 25;
            md.Power = -25;
            md.SendToNXT()
            me.SendToNXT()
        else
            %Stop
            md.Stop('Off')
            me.Stop('Off')
        end
    end
end
```

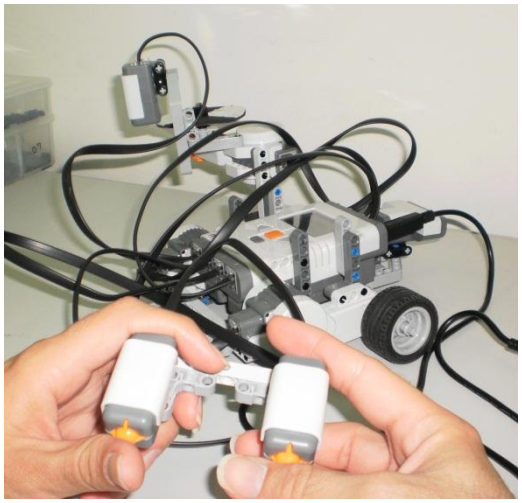

Figura 26 – Programa 2 (*Touch Control)*

Fonte: Autoria Própria

3.5.3. Programa 3 – Gráfico Instantâneo do Sensor de Cor

Neste programa o sensor de cor foi utilizado como sensor de luz para mostrar instantaneamente o valor numérico correspondente à cor que ele está “vendo”. A cor da luz foi configurada para a cor vermelha, mas pode ser modificada para verde ou azul. A Figura 27 mostra o funcionamento do sensor de cor na função *light sensor* enquanto o motor “A” gira o disco bicolor nas cores preta e branca.

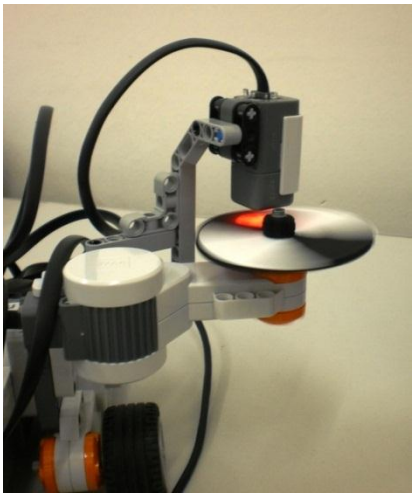

Figura 27 – Programa 3 (Sensor de cor na função *ligth sensor*)

Fonte: Autoria Própria

É possível ver o gráfico sendo gerado instantaneamente devido à presença da função "*pause"* que faz a programação parar por um instante de tempo e depois que o primeiro gráfico é mostrado. Assim, para cada valor do *array* é plotado um gráfico com os valores lidos no instante de tempo t. O código a seguir mostra coma essa combinação pode ser realizada:

```
%%Check toolbox installation: verify that the RWTH - Mindstorms NXT toolbox
is installed.
if verLessThan('RWTHMindstormsNXT', '2.00');
    error(strcat('This program requires the RWTH - Mindstorms NXT Toolbox '
...
        ,'version 3.00 or greater. Go to http://www.mindstorms.rwth-
aachen.de ' ...
        ,'and follow the installation instructions!'));
end%if
%% Clean up: Close previous handles (if existing)
close all
clear all
clc
format compact
%% Iniciando comunicação com o LEGO
h = COM_OpenNXT();
COM_SetDefaultNXT( h )
NXT_PlayTone( 440, 100 )
%% Set up ports
portLuz = SENSOR_3;
%% Ativando Sensor de Cor como um Sensor de Luz
OpenNXT2Color(SENSOR_3, 'RED')
%% Parâmetros
count = 100;
outOfRange = 700;
x = 0 : 1 : count-1 ;
y = zeros(size(x));
%% Ativando Motores
Power = 100;
mA = NXTMotor( MOTOR_A );
%% Gráfico
figure('name', 'Sensor de Luz')
for i = 1 : count
    y(i) = GetNXT2Color(portLuz);
    clf
    hold on
    axis([0 count 0 outOfRange])
    plot(x,y,'m')
    xlabel('Contador/Tempo')
    ylabel('LUZ VALUE')
        mA.Power = Power;
        mA.SendToNXT();
        pause( 0.1 )
    i = i + 1;
end
%% Encerrando comunicação
CloseSensor(portLuz);
mA.Stop('Off');
COM_CloseNXT(h);
```

3.5.4. Programa 4 – Gráfico do Sensor de Cor Utilizando Uma Interface Gráfica

A interface gráfica utilizada é semelhante ao descrito no tópico 3.5.3, mas o gráfico só é mostrado no fim da execução do programa. Enquanto isso, os valores são armazenados nas variáveis definidas em forma de *array*. Esta interface ainda permite a escolha da porta do sensor, da cor emitida pela função *ligth sensor* (Azul, vermelha ou verde, Figura 28), da cor da análise do sinal, da velocidade do disco além de permitir que a imagem por ele gerada seja salva no diretório desejado. A Figura 29 exemplifica a interface gráfica criada.

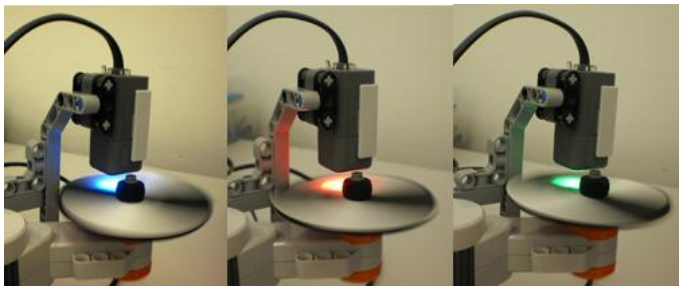

Figura 28 – Programa 4 (Cores da função *light sensor*)

Fonte: Autoria Própria

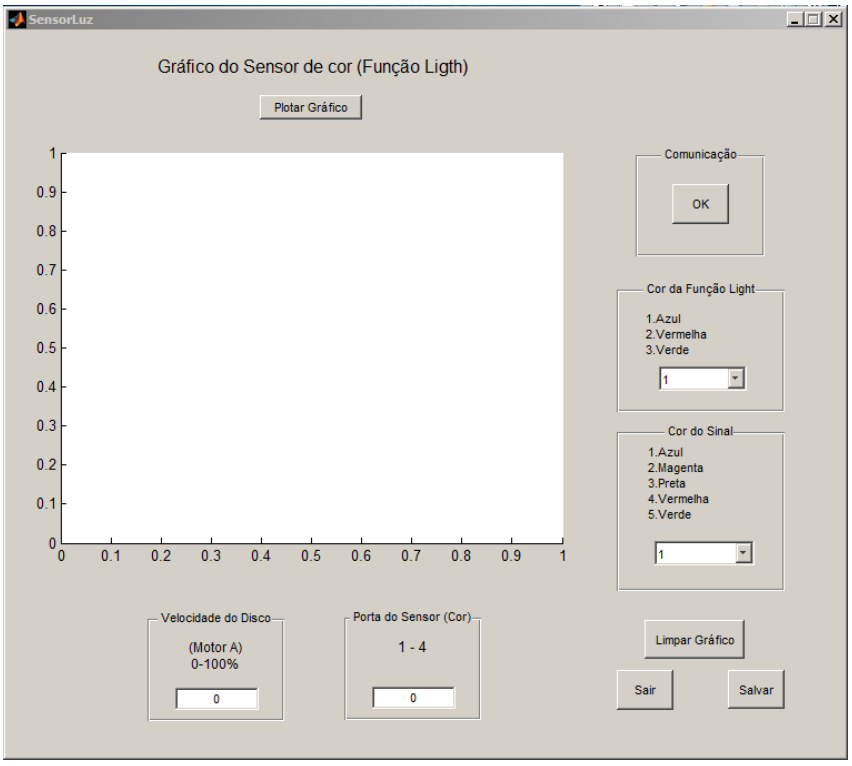

Figura 29 – Programa 4 (Interface Gráfica do *ligth sensor*)

Fonte: Elaboração Própria

O programa desenvolvido para o Programa 4 encontra-se no Anexo B.

3.5.5. Programa 5 – Gráfico do Sensor Ultrassônico

Neste programa há uma mistura de duas funções: manter a distância de um obstáculo e gerar o gráfico instantaneamente da distância "vista" pelo sensor (Figura 30). O programa foi baseado nos exemplos disponíveis pelo *RWTH Mindstorms Toolbox for MATLAB®*. A velocidade, ou potência do motor, é controlada proporcionalmente ao erro, visto que se define uma distância a ser mantida pelo robô e o programa deve controlar sua velocidade de forma a corrigir o erro até ele ser igual a zero. Outros tipos de controle também podem ser utilizados, como o *PID*, por exemplo.

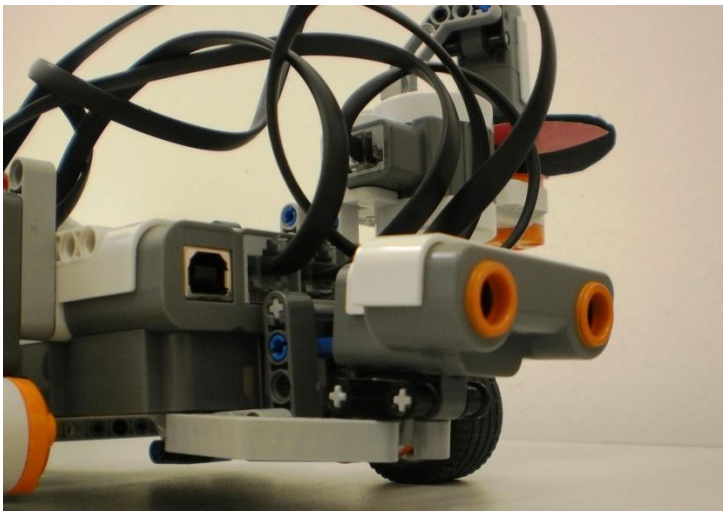

Figura 30 – Programa 5 (Sensor Ultrassônico utilizado para controle da distância)

Fonte: Autoria Própria

O código utilizado para este controle está listado a seguir:

```
% Maintain distance
%%Check toolbox installation: verify that the RWTH - Mindstorms NXT toolbox
is installed.
if verLessThan('RWTHMindstormsNXT', '2.00');
    error(strcat('This program requires the RWTH - Mindstorms NXT Toolbox '
...
        ,'version 3.00 or greater. Go to http://www.mindstorms.rwth-
aachen.de ' ...
        ,'and follow the installation instructions!'));
end%if
```

```
%% Clean up: Close previous handles (if existing)
close all
clear all
clc
format compact
%% Iniciando comunicação com o LEGO
h = COM_OpenNXT();
COM_SetDefaultNXT( h )
NXT_PlayTone( 440, 100 )
%% Prepare graphics
figure('name', 'Next Generation Ultrasound')
set(gca, 'Color', 'black');
hold on
%% Set up ports
portUS = SENSOR_4;
%% Initialize
OpenUltrasonic(portUS, 'snapshot')

n          = 8;          % bytes the US sensor received
count      = 100;        % how many readings until end?
plotcols   = 8;          % how many out of n echos to plot?
outOfRange = 50;        % setting for out of range readings

colors = flipud(hot(8));
data = zeros(1, n);
aux = (1:count+1)';
%% Ativando Motores
m = NXTMotor( [MOTOR_B; MOTOR_C] );
m.SmoothStart = true;
%%
target = 20; % Distância a ser mantida de qualquer obstáculo
    for i = 1 : count
        USMakeSnapshot(portUS)
        pause(0.05);            % wait for the sound to travel
        echos = USGetSnapshotResults(portUS);
        echos(echos == 255) = outOfRange;
        echos = [echos(1); diff(echos)];
        data = vertcat(data, echos');
        x = aux(1:i+1);
        clf
        hold on
        set(gca, 'Color', 'black');
        axis([0 count 0 outOfRange])
        for j = plotcols : -1 : 1
            area(x, data(:, j) , 'FaceColor', colors(j, :))
        end
        % Programação pra manter a distância em 15cm
        d = GetUltrasonic( portUS );
        m.Power = floor( (d - target) * 10);
        m.SendToNXT()
        pause( 0.1 )
    end%for


%% Clean up
CloseSensor(portUS)
COM_CloseNXT(h);
%end
```

3.5.6. Programa 6 – Gráfico do Sensor Ultrassônico Utilizando Uma Interface Gráfica

O programa utilizado na construção desta interface é semelhante ao anterior, mas os valores não podem ser observados instantaneamente. Cada valor lido pelo sensor é armazenado em um *array*, e plotado assim que o contador do programa atinge o limite especificado. Com esta interface gráfica (Figura 31) é possível configurar o *set point*, ou valor desejado, para o motor manter a distância e também se pode modificar porta de entrada do sensor. O programa desta interface gráfica encontra-se no Anexo C.

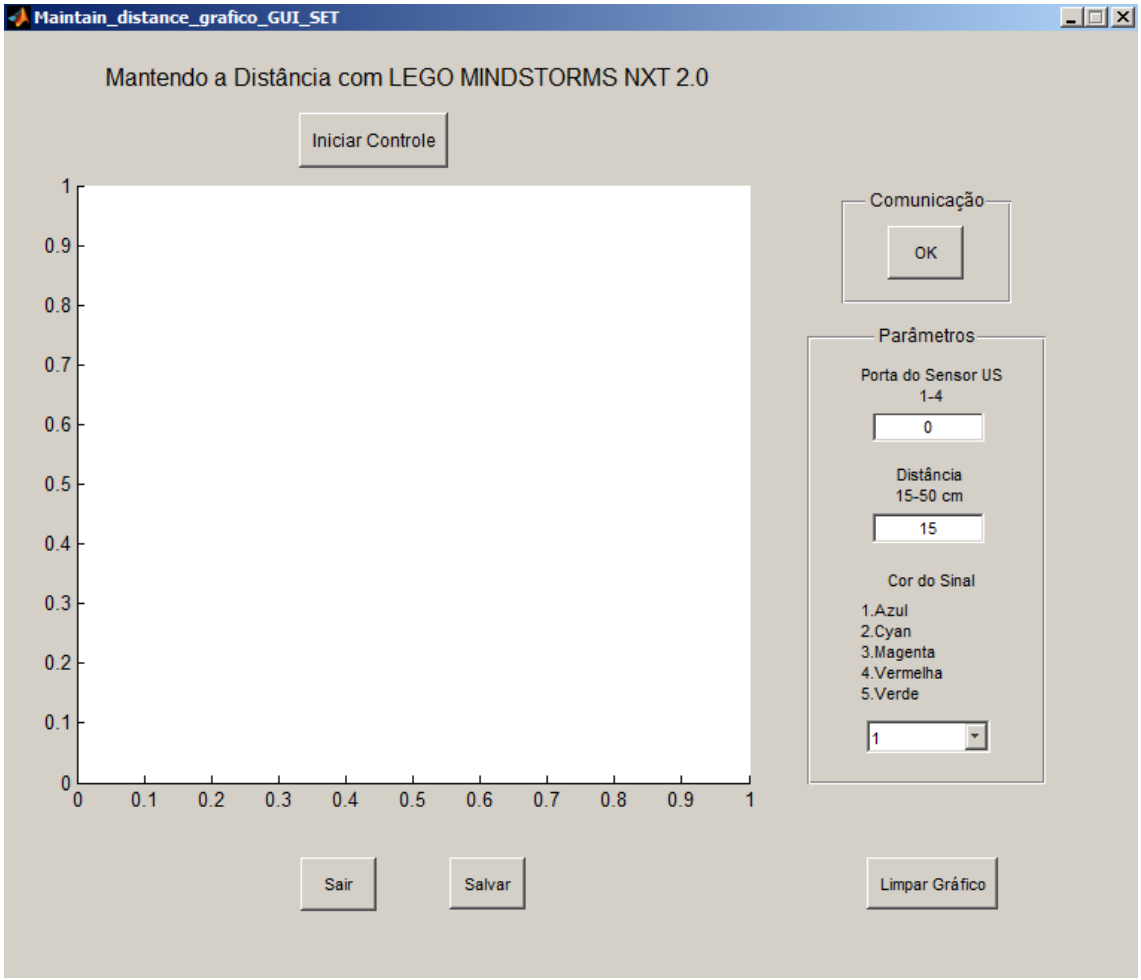

Figura 31 – Programa 6 (Interface Gráfica do sensor ultrassônico)

Fonte: Elaboração Própria

4. RESULTADOS

Através dos programas desenvolvidos no tópico 3.5 podem-se obter gráficos para análise dos sinais enviados pelos sensores. A Figura 32 apresenta o gráfico gerado instantaneamente pelo sensor de cor, na função *light sensor*, utilizando o Programa 3 do tópico 3.5.3.

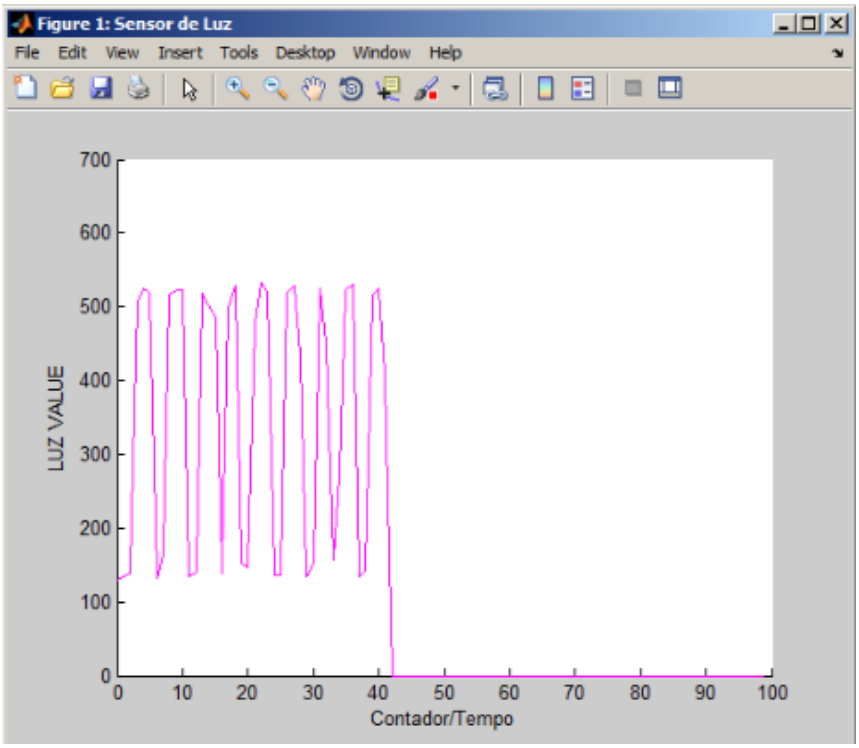

Figura 32 – Gráfico 1 (Valor instantâneo do *ligth sensor* sendo gerado com velocidade do disco em 100 %)

Fonte: Elaboração Própria

O gráfico da Figura 32 foi gerado a partir do disco com cores preta e branca com uma velocidade de 100 % (Figura 27). A faixa de valores gerada para a cor preta está entre 100 e 200, já para a cor branca, os valores variam acima de 500. A Figura 33 mostra outro gráfico gerado com o mesmo programa, mas com uma velocidade de 10 %.

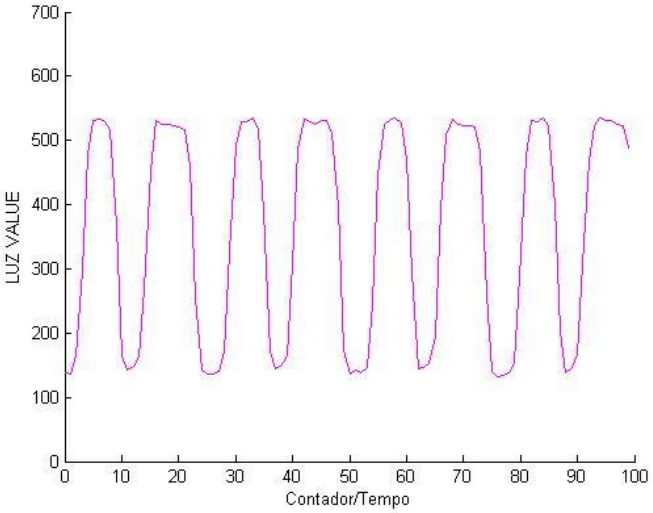

Figura 33 – Gráfico 2 (Valor instantâneo do sensor de cor sendo gerado com velocidade do disco em 10 %.)

Fonte: Elaboração Própria

As Figuras 34, 35 e 36 mostram os valores gerados pelo sensor de cor na função *light sensor* do Programa 4. A cor do sinal foi configurada para luz azul, vermelha e verde, respectivamente. A interface gráfica utilizada se encontra no tópico 3.5.4, as cores do disco utilizado foram preta e prata brilhante (Velocidade de giro de 30 %).

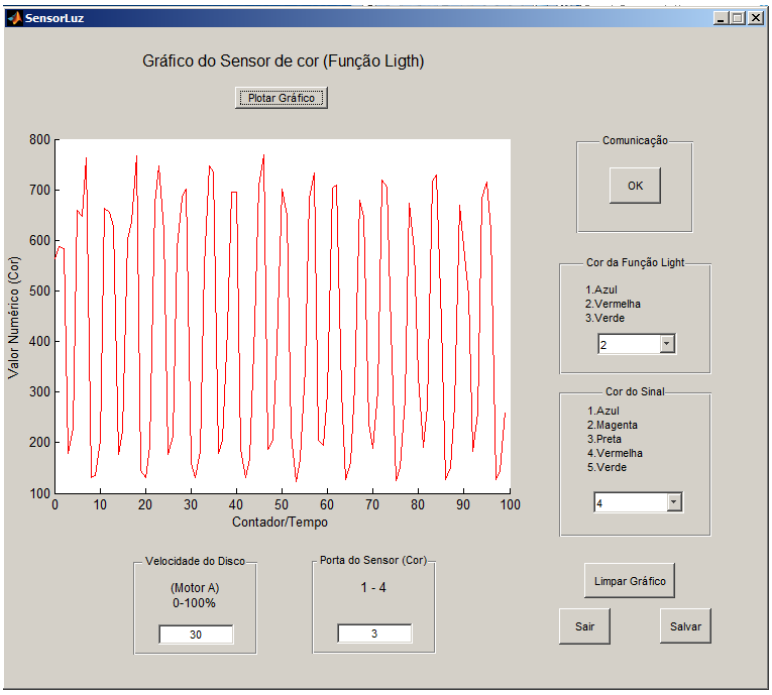

Figura 34 – Gráfico 3 (Programa 4)

Fonte: Elaboração Própria

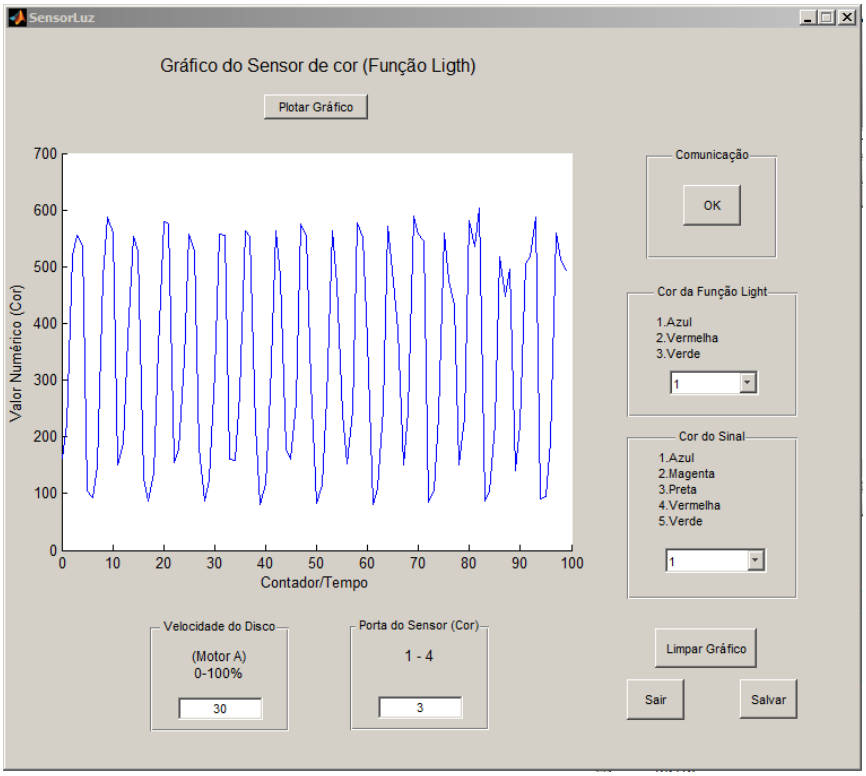

Figura 35 – Gráfico 4 (Programa 4)

Fonte: Elaboração Própria

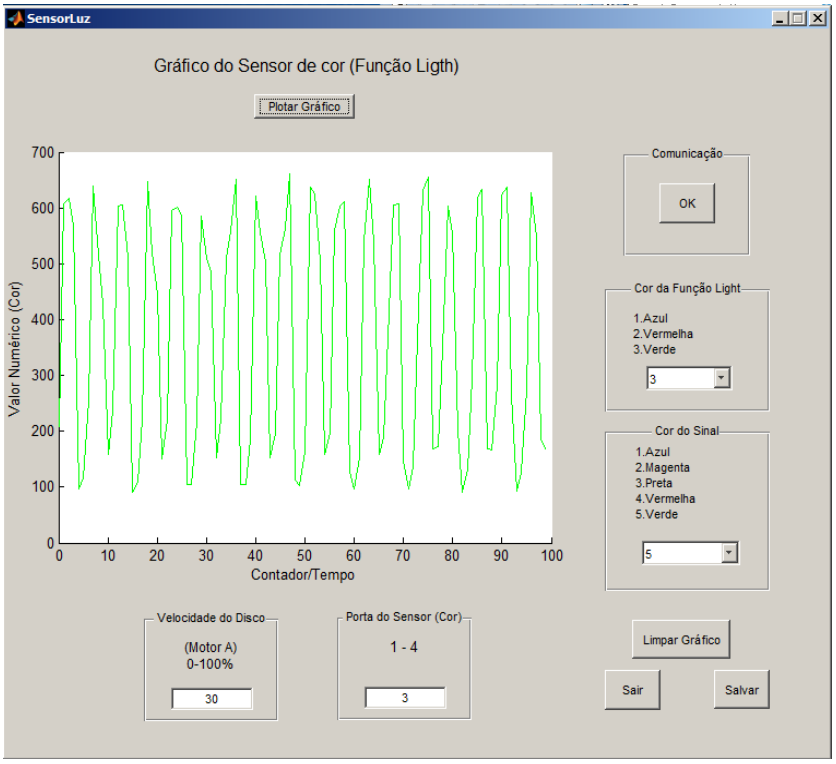

Figura 36 – Gráfico 5 (Programa 4)

Fonte: Elaboração Própria

A Figura 37 apresenta mais um exemplo da interface gráfica anterior, mas com discos de cores (a) preta e azul; (b) preta e amarela; (c) preta e verde e (d) preta e vermelha.

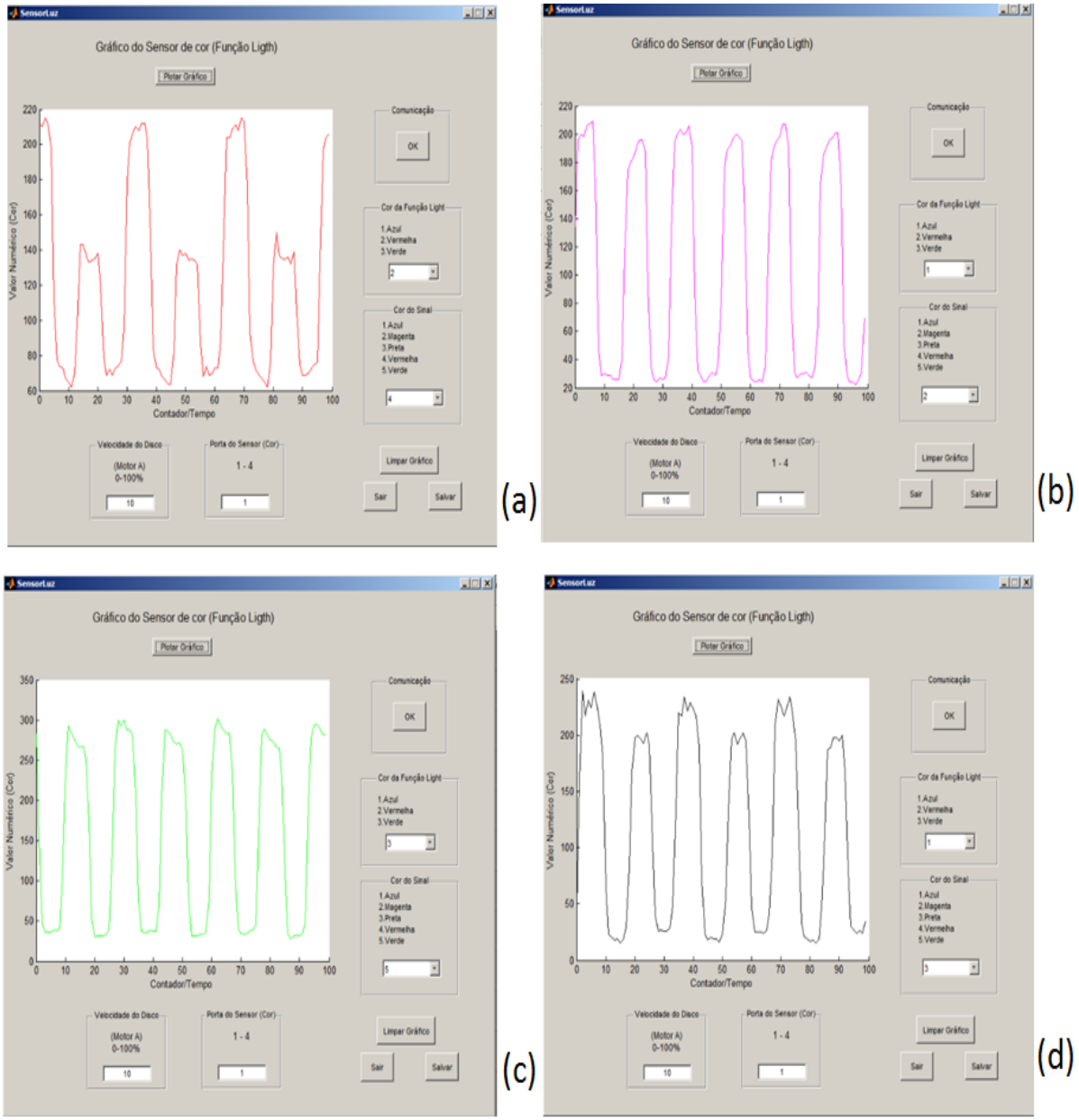

Figura 37 – Gráficos 6.a, 7.b, 8.c e 9.d **(**Programa 4**)**; (b) Disco de cores preta e amarela; (c) Disco de cores preta e verde e (d) Disco de cores preta e vermelha.

Fonte: Elaboração Própria

A Figura 38 mostra o gráfico gerado instantaneamente utilizando o Programa 5 do tópico 3.5.5, que realiza o controle proporcional da potência do motor de acordo o erro calculado entre o *set point* (Valor desejado) e o valor lido pelo sensor ultrassônico, até o

momento em que o robô ficar parada quando o erro for igual a zero, ou seja, a potência igual a zero.

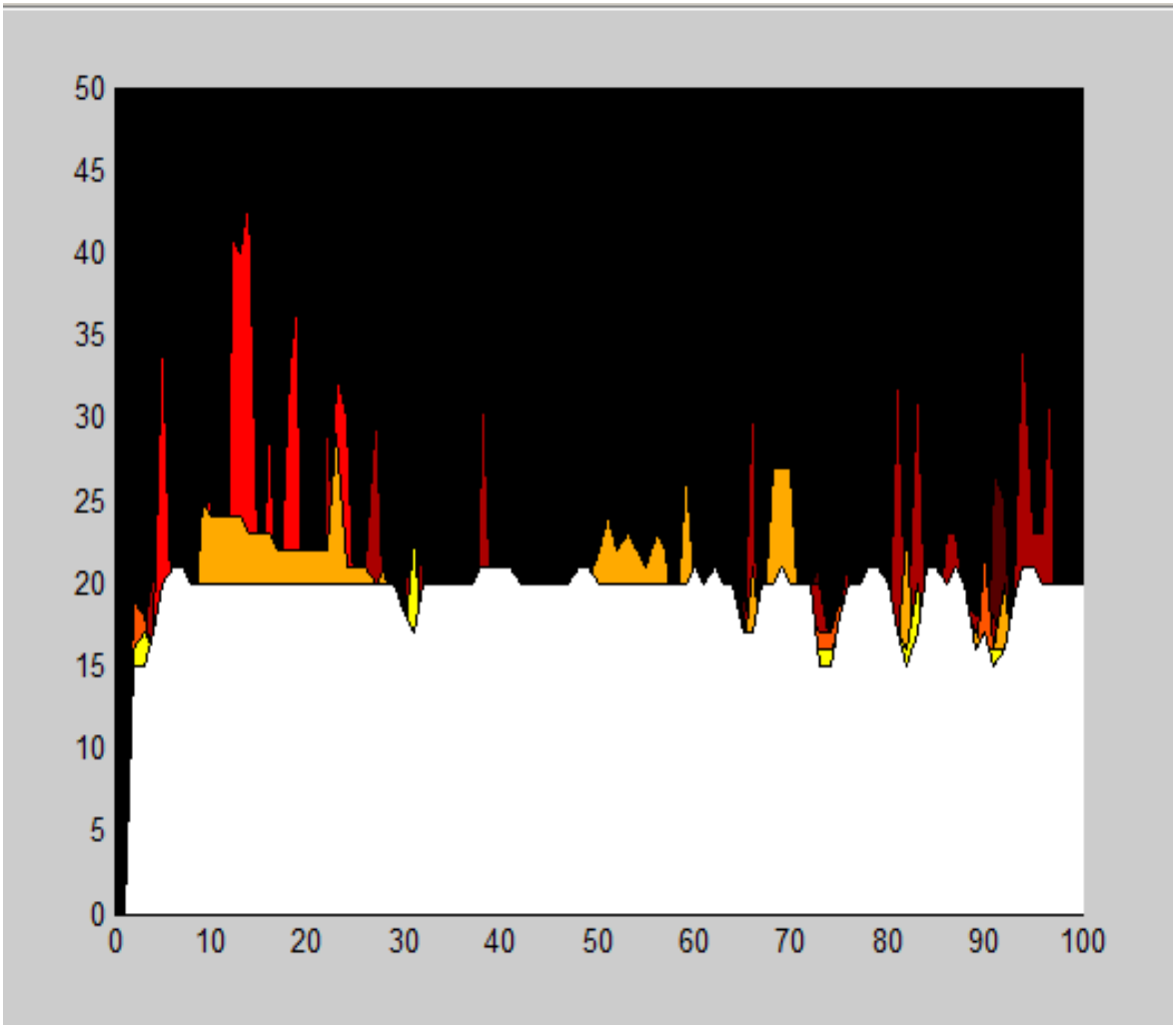

Figura 38 – Gráfico 10 (Distância lida instantaneamente pelo sensor ultrassônico)

Fonte: Elaboração Própria

Com a utilização da interface gráfica gerada através do Programa 6 mostrado no tópico 3.5.6, é possível visualizar os resultados no gráfico gerado e comparar os valores da distância lidos pelo sensor ultrassônico com o *set point* especificado entre 15 e 50 cm. A seguir, nas Figuras 39 e 40 são mostrados dois gráficos com *set points* diferentes e seus respectivos valores de distância lidos.

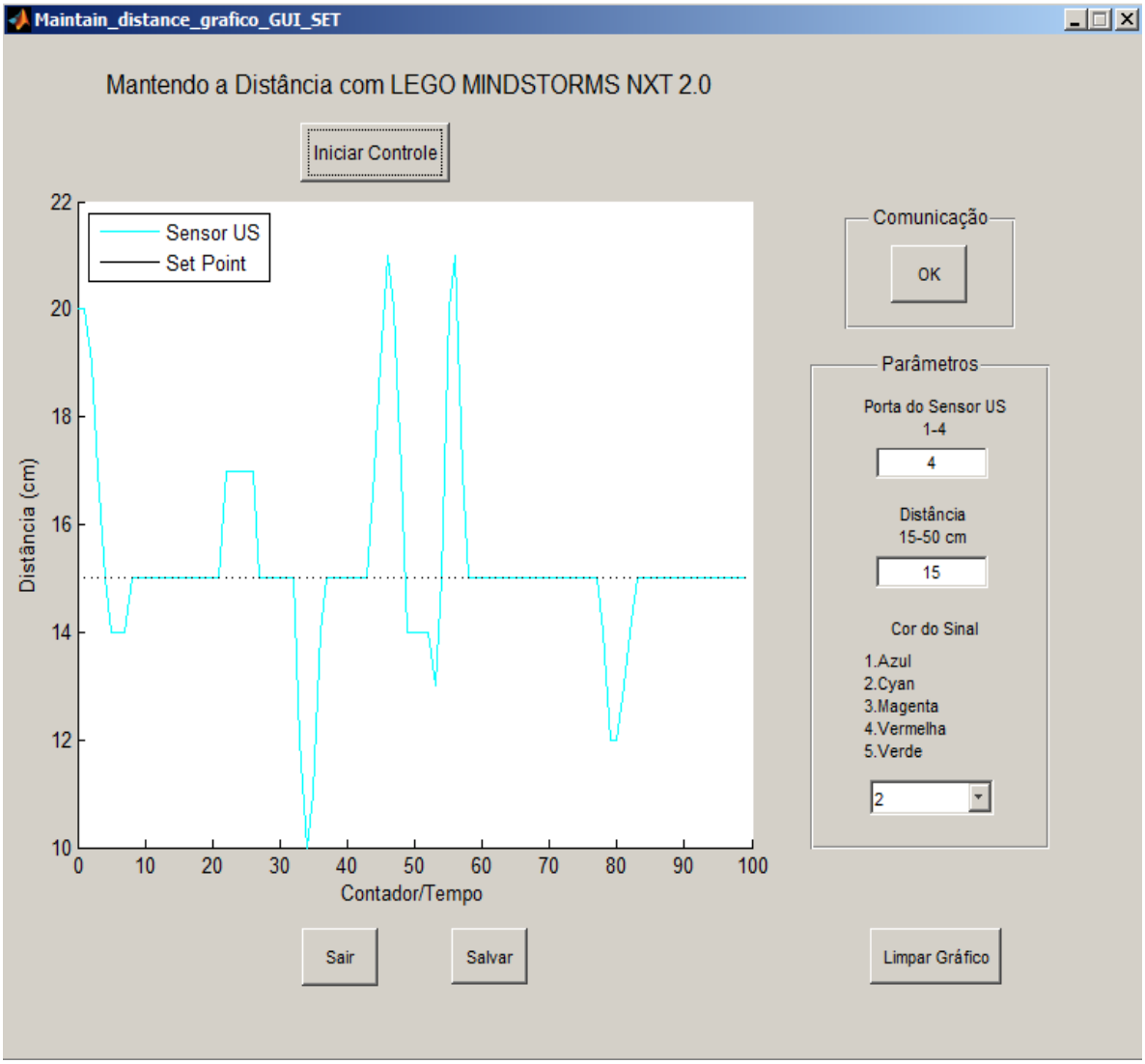

Figura 39 – Gráfico 11 (Controle Proporcional da distância utilizando uma interface gráfica)

Fonte: Elaboração Própria

A variação do obstáculo foi feita de forma aleatória e à medida que se variava a distância, o robô se movimentava tentando permanecer com a distância estabelecida pelo usuário. Na Figura 40 o *set point* especificado foi 20 cm, a porta de *input* utilizada pelo sensor ultrassônico no *smart brick* foi a de número 4 e a cor do sinal escolhida foi a vermelha (Opção de número 4 na interface gráfica).

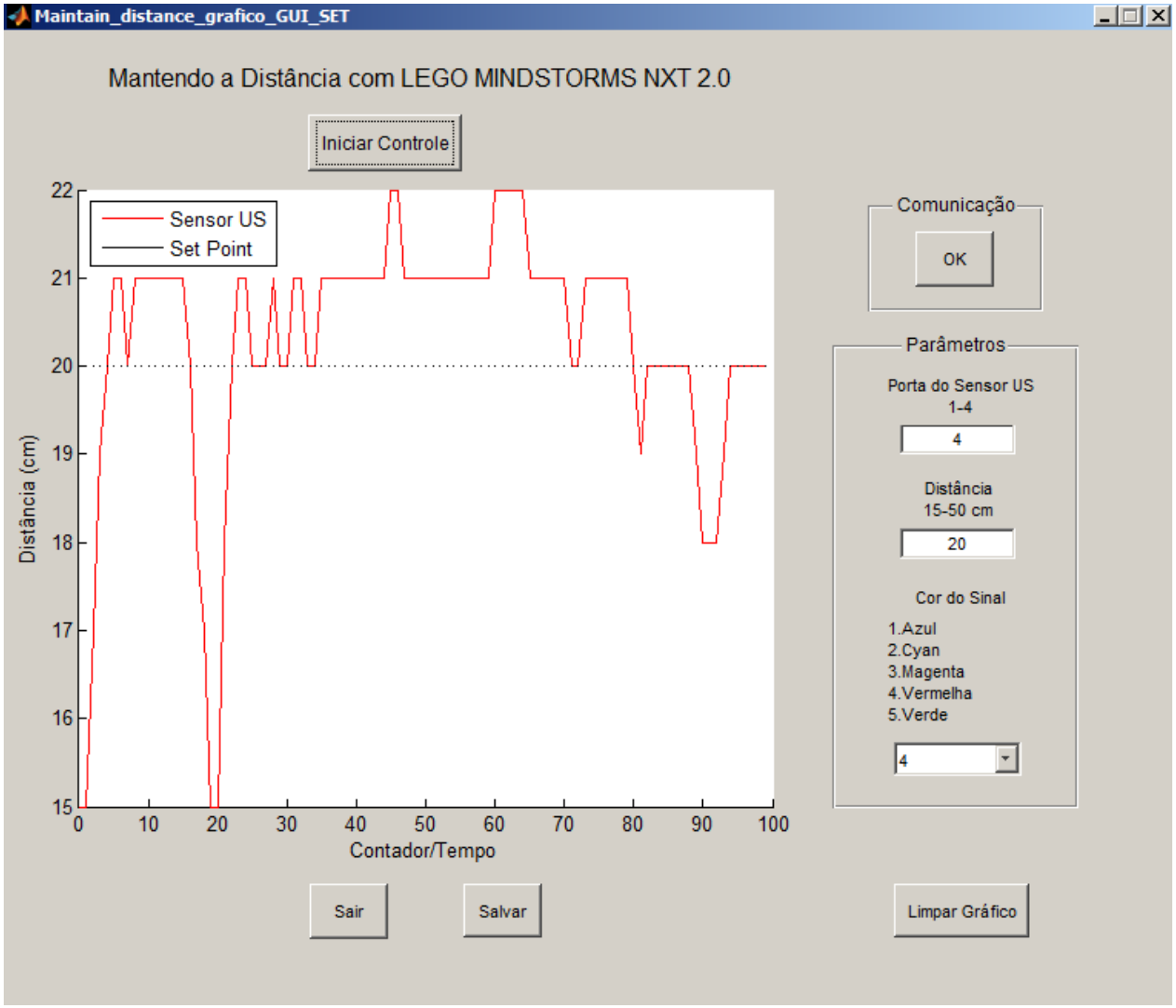

Figura 40 – Gráfico 12 (Controle Proporcional da distância utilizando uma interface gráfica)

Fonte: Elaboração Própria

5. CONSIDERAÇÕES FINAIS

Com o trabalho desenvolvido verificou-se que a programação de *LEGO MINDSTORMS NXT 2.0* utilizando a ferramenta *RWTH – Mindstorms NXT Toolbox for MATLAB®*, é simples e de fácil compreensão. Foram utilizados comandos e conceitos básicos de programação, mas que complementam os conhecimentos de lógica, algoritmos, controle e matemática. O passo a passo desenvolvido para programação descrita no Capítulo 3 pode ser utilizado também como um tutorial na língua portuguesa pelos alunos ingressantes nos primeiros semestres dos cursos de tecnologia e poderia se tornar uma disciplina de pré-requisito para qualquer outra do semestre posterior, visto que a robótica educacional aplica-se a qualquer área de estudo.

O *MATLAB®* torna-se mais atrativo que as outras ferramentas citadas no tópico 2.4 por proporcionar interação com as outras ferramentas disponíveis no mesmo, que possibilitam efetuar cálculos diversos, algoritmos de alta complexidade aplicados à inteligência artificial, analisar os sinais instantaneamente dos dispositivos de entradas e saídas, entre outros citados no tópico 2.3, que são utilizados nos diversos cursos de graduação e pós-graduação.

Almeja-se também desenvolver cursos de introdução à robótica educacional para os alunos ingressantes nos cursos de engenharia e tecnologia da UFERSA, com a ferramenta *RWTH – Mindstorms NXT Toolbox for MATLAB®*, acompanhando o rendimento individual de cada participante além de comparar o desempenho nas demais disciplinas dos cursos com os alunos não participantes. Espera-se ainda trabalhar com os alunos de semestres mais avançados com a intenção de desenvolver protótipos mais elaborados e relacionados às disciplinas mais difíceis dos seus respectivos cursos.

6. REFERÊNCIAS

ACKLAM, Peter J**. MATLAB® *array manipulation tips and tricks*.** Disponível em <http://www.ee.columbia.edu/~marios/MATLAB/MATLAB%20array%20manipulation%20tips%20and%20tricks.pdf>. Acesso em 12 de ago. de 2013.

B, M. C. M. **O Robot NXT Mindstorms e a Área de Projecto**, 2010. 141f. Dissertação (mestrado em Tecnologias de Informação e Comunicação e Educação) – Departamento de sistemas de comunicação, Universidade de Lisboa – PT, 2010. Disponível em: <http://repositorio.ul.pt/bitstream/10451/2491/1/ulfp035871_tm.pdf>. Acesso em: 1 set. 2013.

BARROS, Renata Pitta. **RoboEduc – Uma ferramenta para programação de Robôs LEGO.** Monografia (Engenharia de Computação) – Universidade Federal do Rio Grande do Norte, Natal, 2008.

BEHRENS, Alexander. **MATLAB® meets LEGO MINDSTORMS - *A Freshman Introduction Course Into Practical Engineering***. *IEEE Transactions On Education*, Vol. 53, No. 2, May 2010. Disponível em <http://www.lfb.rwth-aachen.de/files/publications/2010/BEH10e.pdf>. Acesso em 31 de mai. de 2013.

BENITTI, Fabiane BarretoVavassori. *et all*. **Experimentação com Robótica Educativa no Ensino Médio: ambiente, atividades e resultados.** Blumenau-SC, *Universiteit Van Amsterdam.*

BORGES, José. **Computação e Programação**. Disponível em< https://dspace.ist.utl.pt/bitstream/2295/753176/1/AP10.pdf>. Acesso em 10 de ago. de 2013.

BRICX COMMAND CENTER 3.3: **Windows Program.** Sourceforge. USA. Disponível em < http://bricxcc.sourceforge.net/>. Acesso em 18 de setembro de 2013.

CÉSAR, Danilo Rodrigues. BONILLA, Maria Helena Silveira. **Robótica Livre: Implementação de um ambiente dinâmico de robótica pedagógica com soluções tecnológicas livres no Cet CEFET em Itabirito – Minas Gerais – Brasil.** Salvador-BA. Universidade Federal da Bahia (UFBA). 2007.

FERRI, Bonnie Heck. *et all.* ***Signal Processing with the LEGO MINDSTORMS NXT Kit for Use in Signals and Systems Courses.*** St. Louis, MO-USA. American Control Conference. 2009.

FLEURY, Cláudio A. **Guide Interface Gráfica para Aplicações MATLAB®**. Disponível em<http://professor.ucg.br/SiteDocente/admin/arquivosUpload/5380/material/Guide.pdf>. Acesso em 10 de ago. de 2013.

Gnu- *General Public License*.Disponível em<http://www.gnu.org/licenses/gpl-howto.html>. Acesso em 7de jul. de 2013.

GOMES, Marcelo Carboni. *et all.* **KickRobot: Inclusão digital através da robótica em escolas públicas do Rio Grande do Sul.** Porto Alegre-RS. Universidade Federal do Rio Grande do Sul (UFRGS). Maravilha-SC. Universidade do Oeste de Santa Catarina (UNOESC.). 2008.

GROCHOCKI, Luiz Rodrigo. SILVA, Rodrigo Barbosa E. **Robótica Educacional.** Guarapuava-PR.

HANSELMAN, Duane. LITTLEFIELD, Bruce. **Matlab®:** Curso Completo. São Paulo: Pearson, 2004.676p.

Install Lego USB Driver: Video Instructions. RWTH Toolbox Installation, United States, Ago. 2009. Disponível em: < http://people.clemson.edu/~nwatts/engr141/instructions/>. Acesso em: 11 jul. 2013.

L, J. **O fazer do trabalho cientifico em ciências sociais aplicadas.** 1. ed. Recife: Universitária da UFPE, 2006. 303p.

LEGO EDUCATION: LEGO Factory. **LEGO do Brasil**, São Paulo, mar. 2013. Disponível em: <http://legodobrasil.com.br/products/6633-lego-education-ev3-core-set-frete-gratis.aspx>. >. Acesso em: 1 set. 2013.

LEGO EDUCATION. Disponível em: < http://www.legoeducation.us/eng/product/lego_mindstorms_education_nxt_base_set/2095>. Acesso em: 20 set. 2013.

LEGO MINDSTORMS NXT 2.0. Disponível em < http://mindstorms.lego.com/>. Acesso em 31 de mai. de 2013.

LEGO MINDSTORMS NXT 2.0: **Amazom.** Disponível em < http://www.amazon.com/gp/product/B001USHRYI?ie=UTF8&tag=nxtprogramsco-20&linkCode=as2&camp=1789&creative=9325&creativeASIN=B001USHRYI>. Acesso em 20 de satembro de 2013.

LEGO MINDSTORMS NXT: Hardware Support. **Mathworks**, United States, Ago. 2009. Disponível em: <http://www.mathworks.com/hardware-support/lego-mindstorms-MATLAB®.html>. Acesso em: 11 jul. 2013.

LEGO, **LEGO MINDSTORES User Guide**, International, 2009.

LEGO, LEGO Mindstorms NXT. Disponível e <http://www.nxtprograms.com/index.html>. Acesso em 31 de mai. de 2013.

L, J. **O fazer do trabalho cientifico em ciências sociais aplicadas.** 1. ed. Recife: Universitária da UFPE, 2006. 303p.

M, F. N. Um simulador de robótica para MATLAB® (mini tutorial). **Nossos Robôs**, São Paulo, mar. 2013. Disponível em: < http://nossosrobos.blogspot.com.br/2013/03/robotics-toolbox-um-simulador-de.html>. Acesso em: 13 jul. 2013.

MATHWORKS, **Centro de Documentação**. Disponível em<http://www.mathworks.com/help/MATLAB/creating_guis/gui-with-multiple-axes-guide.html>. Acesso em 14 de ago. de 2013.

MATLAB® ANSWERS: ask question. **MATLAB® Central**, United States, jul. 2013. Disponível em: < http://www.mathworks.com/MATLABcentral/answers/36010-save-figure-from-gui-with-axes-and-colorbar>. Acesso em: 15 ago. 2013.

MATLAB® – ***The language of technical computing***. Disponível em <http://www.mathworks.com/products/datasheets/pdf/MATLAB.pdf> Acesso em 02 de jun. de 2013.

MATLAB® Meets LEGO Mindstorms . ***A Freshman Introduction Course Into Practical Engineering***. Disponível em<http://www.lfb.rwth-aachen.de/files/publications/2010/BEH10e.pdf>. Acesso em 31 de mai. de 2013.

MATLAB® *Support for* LEGO Mindstorms NXT. Disponível em <http://www.mathworks.com/tagteam/65410_91837v00_LEGOMindstormsML_FactSheet_8-5x11_final.pdf> Acesso em 31 de mai. de 2013.

MATLAB® *the linguagen of technical computing*. Disponível em<http://www.mathworks.com/products/datasheets/pdf/MATLAB.pdf>. Acesso em 2 de jun. de 2013.

MATLAB®, *Basics of Plotting in MATLAB®*. Disponível em<**http://acmg.seas.harvard.edu/forum_files2/gsf_MATLAB_plotting.pdf**>. Acesso em 12 de ago. de 2013.

MATLAB®, **Minicurso Rápido e Prático**. Disponível em < http://www.densis.fee.unicamp.br/~anibal/MATLABMan/Cap7.doc>. Acesso em 10 de ago. de 2013.

MATLAB®-7.Disponível em<http://www.ebah.com.br/content/ABAAAAoVsAG/MATLAB-7#>. Acesso em 10 ago. de 2013.

MATSUMOTO, Élia Yathie. **MATLAB® 7:** Fundamentos. 2. ed. São Paulo: Érica, 2008. 376p.

MINDSTORMS NXT: GNU General Public License. **RWTH**, United States, Jan. 2008. Disponível em: < http://www.mindstorms.rwth-aachen.de/trac/wiki/GPL>. Acesso em: 11 jul. 2013.

MINDSTORMS NXT: MATLAB® toolbox. **RWTH**, United States, Jan. 2008. Disponível em: < http://www.mindstorms.rwth-aachen.de/trac/wiki/Download>. Acesso em: 11 jul. 2013.

MINDSTORMS UPDATE: Next tools. **Programmable Brick Utilities**, United States, Ago. 2010. Disponível em: < http://bricxcc.sourceforge.net/utilities.html>. Acesso em: 11 jul. 2013.

MINDSTORMS NXT: Toolbox documentation. **RWTH**, United States, jan. 2008. Disponível em: < http://www.mindstorms.rwth-aachen.de/trac/wiki/Download>. Acesso em: 11 jul. 2013.

MUNARO, C. **Manual do MATLAB®:** Curso de MATLAB®. Espírito Santo: Laboratório de Computação da Engenharia Elétrica – UFES, 1997.

NETO, Geraldo Furtado. *et al* **. Utilização do KIT LEGO MINDSTORM NXT no ensino de controle de processos**. XL Congresso Brasileiro de Educação em Engenharia, 2012. Disponível em < http://www.abenge.org.br/CobengeAnteriores/2012/artigos/104237.pdf>. Acesso em 2 de jun. de 2013.

OFFICE 2010: Instructions. **PowerPoint**, United States, abr. 2010. Disponível em: < http://people.clemson.edu/~nwatts/engr141/instructions/PP_Narration.pdf>. Acesso em: 11 jul. 2013.

OPERATING SYSTEM: GNU General Public License. **GNU**, United States, jul. 2013. Disponível em: <http://www.gnu.org/licenses/why-not-lgpl.en.html>. Acesso em: 11 jul. 2013.

PESTANA, HÉLDER GOLVEIA. **DROID M.L.P. NXT *Software* Development KIT**. Disponível em <http://dme.uma.pt/projects/droide/portal/DROIDE MLP NXT *Software* Development kit.pdf>. Acesso em 31 de mai. de 2013.

REQUIRED EQUIPMENT: Lego Mindstorms NXT. **RWTH**, United States, jan. 2008. Disponível em: <http://www.clemson.edu/ces/departments/ece/document_resource/undergrad/MATLAB/Lab%20Manual.pdf>. Acesso em: 11 jul. 2013.

ROSSATO, Daniel B. **Desenvolvimento de um sistema aberto para ensino de >robótica**. Disponível em < http://www.labplan.ufsc.br/congressos/CBA2008/textos/CBA_2008_Artigos/41256.pdf> Acesso em 02 de jun. de 2013.

RWTH- *Mindstorms NXT Toolbox*. Disponível em<http://www.mindstorms.rwth-aachen.de/trac/wiki/Download>. Acesso em 7 de jul. de 2013.

S, A. J. **Metodologia do Trabalho Científico.** 23. ed. São Paulo: Cortez, 2007. 299p.

SCHNEIDER, David. ***A droid for all seasons***. *IEEE spectrum, International,* v. 48, n. 12, p. 20-22, dez.2011.

SIQUEIRA, Jan Krueger. **Guia MATLAB® para alunos de Engenharia Elétrica, Computação e Automação**. Disponível em < http://www2.ic.uff.br/~aconci/GuiaMATLAB.pdf>. Acesso em 10 de ago. de 2013.

VIEGAS, Carlos Emanuel Dias. **Uma plataforma de Comunicação para Robôs Pioneer AT e LEGO Mindstorms RCX.** 57 f. Monografia (Engenharia de Computação) – Universidade Federal do Rio Grande do Norte, Natal, 2012.

VIEIRA, Danilo R**. Criando uma interface gráfica para obter distâncias entre pontos em uma figura.** Disponível em <http://www.danilorvieira.com/MATLAB/gui-ptos/#3>. Acesso em 10 de ago. de 2013.

YATHIE, E. M. **MATLAB® 7 – FUNDAMENTOS.** 1. ed. São Paulo, Cortez, 2004. 380p.

GLOSSÁRIO

ARRAY	Organizador de parâmetros
ADD FOLDER	Adicionar pasta
BUMPED	Colidir
COLOR LAMP	Cor da lâmpada
COLOR SENSOR	Sensor de Cor
CONTINUOUS MODE	Modo contínuo
DEMOS	Demonstração
EDUCATION	Educação
ELECTRICAL ENGINEERING	Engenharia Elétrica
ELECTRICAL FLOW	Fluxo / corrente elétrica
FIVE MINUTE BOT	Robô *LEGO* que pode ser construído em cinco minutos
HANDLE	Espécie de “alça” que guarda as características do objeto
HARDWARE	Equipamento
HOME	Início
INFORMATION TECHNOLOGY	Tecnologia da Informação
INPUTS	Dispositivos de entrada
INSTITUTE OF IMAGING & COMPUTER VISION	Instituto de Imagem e Visão Computacional
LEGO GROUP	Grupo *LEGO*
LIGHT SENSOR	Sensor de Luz
MOTORCONTROL.RXE	Controle dos motores / robôs
OBJECT	Objeto
OUTPUTS	Dispositivos de Saída
PAUSE	Pausa
PID	Controle Proporcional, Integral e Derivativo
PLC	Controlador Lógico Programável
PROTÓTIPOS	Robôs

RECEIVER	Recebedor
RGB	Cores aditivas vermelha, verde e azul
ROTATION SENSOR	Sensor de Rotação
RWTH – MINDSTORMS TOOLBOX FOR MATLAB®	Caixa de ferramenta do *MATLAB®* para utilizar com os *kits LEGO MINDSTORMS*
SAVE	Salvar
SENDER	Remetente
SERIAL PORT	Porta Serial
SERVO MOTOR	Máquina que apresenta movimento proporcional a um comando
SET PATH	Selecionar pasta
SET POINT	Valor desejado ou valor a ser atingido
SMART BRICK	Tijolo / Bloco inteligente
SNAPSHOT MODE	Modo instantâneo
SOFTWARE	Programa de Computador
TACHOMETER	Tacômetro
TECHNICAL COMPUTER SCIENCE	Técnico de Informática
TOOLBOX	Caixa de ferramentas
TOOLS	Ferramentas
TOUCH CONTROL	Controle de toque
ULTRASSÔNICO	Onda Sonora de alta frequência
UNIVERSITY	Universidade
VALUE	Valor
VOLTAGE MEASURMENT	Medição de Tensão
WIRELESS BLUETOOTH	*Bluetooth* sem fio

ANEXO A – Programa 1 (Controle Simples Utilizando uma Interface Gráfica)

```
function varargout = ControleSimplesLego(varargin)
% CONTROLESIMPLESLEGO MATLAB® code for ControleSimplesLego.fig
%      CONTROLESIMPLESLEGO, by itself, creates a new CONTROLESIMPLESLEGO or
raises the existing
%      singleton*.
%
%      H = CONTROLESIMPLESLEGO returns the handle to a new
CONTROLESIMPLESLEGO or the handle to
%      the existing singleton*.
%
%      CONTROLESIMPLESLEGO('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in CONTROLESIMPLESLEGO.M with the given
input arguments.
%
%      CONTROLESIMPLESLEGO('Property','Value',...) creates a new
CONTROLESIMPLESLEGO or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before ControleSimplesLego_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to ControleSimplesLego_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help ControleSimplesLego

% Last Modified by GUIDE v2.5 06-Sep-2013 00:46:36

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ControleSimplesLego_OpeningFcn, ...
                   'gui_OutputFcn',  @ControleSimplesLego_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before ControleSimplesLego is made visible.
function ControleSimplesLego_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ControleSimplesLego (see VARARGIN)

% Choose default command line output for ControleSimplesLego
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes ControleSimplesLego wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = ControleSimplesLego_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in robEsq.
function robEsq_Callback(hObject, eventdata, handles)

h = COM_OpenNXT(); %Estabelecendo comunicação com o LEGO
COM_SetDefaultNXT( h );

potBC = get(handles.rbPotBC, 'String');

mB = NXTMotor(MOTOR_B);
mC = NXTMotor(MOTOR_C);

    mB.Power = -(str2num(potBC));
    mC.Power = (str2num(potBC));
    mB.SendToNXT();
    mC.SendToNXT();

guidata(hObject, handles);

% hObject    handle to robEsq (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in robFre.
function robFre_Callback(hObject, eventdata, handles)

h = COM_OpenNXT(); %Estabelecendo comunicação com o LEGO
```

```
COM_SetDefaultNXT( h );

potBC = get(handles.rbPotBC, 'String');

mB = NXTMotor(MOTOR_B);
mC = NXTMotor(MOTOR_C);

    mB.Power = (str2num(potBC));
    mC.Power = (str2num(potBC));
    mB.SendToNXT();
    mC.SendToNXT();

guidata(hObject, handles);
% hObject    handle to robFre (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in robRe.
function robRe_Callback(hObject, eventdata, handles)

h = COM_OpenNXT(); %Estabelecendo comunicação com o LEGO
COM_SetDefaultNXT( h );

potBC = get(handles.rbPotBC, 'String');

mB = NXTMotor(MOTOR_B);
mC = NXTMotor(MOTOR_C);

    mB.Power = -(str2num(potBC));
    mC.Power = -(str2num(potBC));
    mB.SendToNXT();
    mC.SendToNXT();

guidata(hObject, handles);

% hObject    handle to robRe (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in robDir.
function robDir_Callback(hObject, eventdata, handles)

h = COM_OpenNXT(); %Estabelecendo comunicação com o LEGO
COM_SetDefaultNXT( h );

potBC = get(handles.rbPotBC, 'String');

mB = NXTMotor(MOTOR_B);
mC = NXTMotor(MOTOR_C);

    mB.Power = (str2num(potBC));
    mC.Power = -(str2num(potBC));
    mB.SendToNXT();
    mC.SendToNXT();

guidata(hObject, handles);
```

```
% hObject    handle to robDir (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)



function rbPotBC_Callback(hObject, eventdata, handles)
% hObject    handle to rbPotBC (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)
potBC = str2num(get(hObject,'String'));
if (isempty(potBC))
    set(hObject, 'String','0')
    msgbox('Digite um valor numérico…', 'Atenção');
else
    if potBC < 0 || potBC > 100
        set(hObject, 'String','0')
        msgbox('Digite um valor entre 0 e 100…', 'Atenção');
    else
       guidata(hObject, handles);
    end
end
% Hints: get(hObject,'String') returns contents of rbPotBC as text
%        str2double(get(hObject,'String')) returns contents of rbPotBC as a
double


% --- Executes during object creation, after setting all properties.
function rbPotBC_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rbPotBC (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in rbHor.
function rbHor_Callback(hObject, eventdata, handles)

h = COM_OpenNXT(); %Estabelecendo comunicação com o LEGO
COM_SetDefaultNXT( h );

potA = get(handles.rbPotA, 'String');

mA = NXTMotor(MOTOR_A);

    mA.Power = -(str2num(potA));
    mA.SendToNXT();

guidata(hObject, handles);

% hObject    handle to rbHor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)
```

```
% --- Executes on button press in rbAntH.
function rbAntH_Callback(hObject, eventdata, handles)
h = COM_OpenNXT(); %Estabelecendo comunicação com o LEGO
COM_SetDefaultNXT( h );

potA = get(handles.rbPotA, 'String');

mA = NXTMotor(MOTOR_A);

    mA.Power = (str2num(potA));
    mA.SendToNXT();

guidata(hObject, handles);
% hObject    handle to rbAntH (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in rbParar1.
function rbParar1_Callback(hObject, eventdata, handles)
% hObject    handle to rbParar1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)

h = COM_OpenNXT();%Estabelecendo comunicação com o LEGO
COM_SetDefaultNXT( h );

mB = NXTMotor(MOTOR_B);
mC = NXTMotor(MOTOR_C);

mB.Stop('Off');
mC.Stop('Off');

guidata(hObject, handles);

% --- Executes on button press in rbParar2.
function rbParar2_Callback(hObject, eventdata, handles)
% hObject    handle to rbParar2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)

h = COM_OpenNXT();%Estabelecendo comunicação com o LEGO
COM_SetDefaultNXT( h );

mA = NXTMotor(MOTOR_A);

mA.Stop('Off');

guidata(hObject, handles);


function rbPotA_Callback(hObject, eventdata, handles)
% hObject    handle to rbPotA (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)
potA = str2num(get(hObject,'String'));
if (isempty(potA))
```

```
    set(hObject, 'String','0')
    msgbox('Digite um valor numérico…', 'Atenção');
else
    if potA < 0 || potA > 100
        set(hObject, 'String','0')
        msgbox('Digite um valor entre 0 e 100…', 'Atenção');
    else
       guidata(hObject, handles);
    end

end
guidata(hObject, handles);
% Hints: get(hObject,'String') returns contents of rbPotA as text
%        str2double(get(hObject,'String')) returns contents of rbPotA as a
double


% --- Executes during object creation, after setting all properties.
function rbPotA_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rbPotA (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in rbSair.
function rbSair_Callback(hObject, eventdata, handles)
% hObject    handle to rbSair (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)
s=questdlg('Deseja sair?','Confirmação','Sim','Não');

if s == 'Sim'

    close all;

    mA.Stop('Off');
    mB.Stop('Off');
    mC.Stop('Off');
    COM_CloseNXT(h);

    clear all
    clc
end
guidata(hObject, handles);


% --- Executes on button press in rbCom.
function rbCom_Callback(hObject, eventdata, handles)
% hObject    handle to rbCom (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)
%h = COM_OpenNXT()
```

```
%COM_SetDefaultNXT( h )
%NXT_PlayTone( 440, 100 )
%% Iniciando comunicação com o LEGO
%Check toolbox installation: verify that the RWTH - Mindstorms NXT toolbox
is installed.
if verLessThan('RWTHMindstormsNXT', '2.00');
    error(strcat('This program requires the RWTH - Mindstorms NXT Toolbox '
...
        ,'version 3.00 or greater. Go to http://www.mindstorms.rwth-
aachen.de ' ...
        ,'and follow the installation instructions!'));
end%if

% Clean up: Close previous handles (if existing)
%close all
clear all
clc
format compact

% Iniciando comunicação com o LEGO
h = COM_OpenNXT();
COM_SetDefaultNXT( h )
NXT_PlayTone( 440, 100 )

% Ativando Motores
mA = NXTMotor( MOTOR_A );
mB = NXTMotor( MOTOR_B );
mC = NXTMotor( MOTOR_C );
%FIM PROGRAMA
```

ANEXO B – Programa 4 (Gráfico do Sensor de Cor Utilizando uma Interface Gráfica)

```
function varargout = SensorLuz(varargin)
% SENSORLUZ MATLAB® code for SensorLuz.fig
%      SENSORLUZ, by itself, creates a new SENSORLUZ or raises the existing
%      singleton*.
%
%      H = SENSORLUZ returns the handle to a new SENSORLUZ or the handle to
%      the existing singleton*.
%
%      SENSORLUZ('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SENSORLUZ.M with the given input
arguments.
%
%      SENSORLUZ('Property','Value',...) creates a new SENSORLUZ or raises
the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before SensorLuz_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to SensorLuz_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help SensorLuz

% Last Modified by GUIDE v2.5 04-Sep-2013 23:34:31

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @SensorLuz_OpeningFcn, ...
                   'gui_OutputFcn',  @SensorLuz_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before SensorLuz is made visible.
function SensorLuz_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)
```

```
% varargin   command line arguments to SensorLuz (see VARARGIN)

% Choose default command line output for SensorLuz
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes SensorLuz wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = SensorLuz_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in slCom.
function slCom_Callback(hObject, eventdata, handles)
% hObject    handle to slCom (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)
%% Iniciando comunicação com o LEGO
%Check toolbox installation: verify that the RWTH - Mindstorms NXT toolbox
is installed.
if verLessThan('RWTHMindstormsNXT', '2.00');
    error(strcat('This program requires the RWTH - Mindstorms NXT Toolbox '
...
        ,'version 3.00 or greater. Go to http://www.mindstorms.rwth-
aachen.de ' ...
        ,'and follow the installation instructions!'));
end%if

% Clean up: Close previous handles (if existing)
%close all
clear all
clc
format compact

% Iniciando comunicação com o LEGO
h = COM_OpenNXT();
COM_SetDefaultNXT( h );
NXT_PlayTone( 440, 100 )




function slPortSensor_Callback(hObject, eventdata, handles)
% hObject    handle to slPortSensor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)
portSensor = str2num(get(hObject,'String'));
if (isempty(portSensor))
```

```
    set(hObject, 'String','0')
    %msgbox('Digite um valor numérico…', 'Atenção');
    errordlg('Por favor, digite valor numérico.', 'Erro');
else
    if portSensor < 1 || portSensor > 4
        set(hObject, 'String','0')
        %msgbox('Digite um valor entre 0 e 100…', 'Atenção');
        errordlg('Por favor, digite um número válido.', 'Erro');
    else
       guidata(hObject, handles);
    end
end
guidata(hObject, handles);

% Hints: get(hObject,'String') returns contents of slPortSensor as text
%        str2double(get(hObject,'String')) returns contents of slPortSensor
as a double


% --- Executes during object creation, after setting all properties.
function slPortSensor_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slPortSensor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in slCorSensor.
function slCorSensor_Callback(hObject, eventdata, handles)
% hObject    handle to slCorSensor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns slCorSensor
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
slCorSensor


% --- Executes during object creation, after setting all properties.
function slCorSensor_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slCorSensor (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in slCorSinal.
function slCorSinal_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to slCorSinal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns slCorSinal
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
slCorSinal


% --- Executes during object creation, after setting all properties.
function slCorSinal_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slCorSinal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function slVelDisc_Callback(hObject, eventdata, handles)
% hObject    handle to slVelDisc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)
velA = str2num(get(hObject,'String'));
if (isempty(velA))
    set(hObject, 'String','0')
    %msgbox('Digite um valor numérico…', 'Atenção');
    errordlg('Por favor, digite valor numérico.', 'Erro');
else
    if velA < 0 || velA > 100
        set(hObject, 'String','0')
        %msgbox('Digite um valor entre 0 e 100…', 'Atenção');
        errordlg('Por favor, digite um número válido.', 'Erro');
    else
       guidata(hObject, handles);
    end
end
guidata(hObject, handles);

% Hints: get(hObject,'String') returns contents of slVelDisc as text
%        str2double(get(hObject,'String')) returns contents of slVelDisc as
a double


% --- Executes during object creation, after setting all properties.
function slVelDisc_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slVelDisc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in slPlot.
function slPlot_Callback(hObject, eventdata, handles)
% hObject    handle to slPlot (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)

%% Iniciando comunicação com o LEGO
h = COM_OpenNXT();
COM_SetDefaultNXT( h );

%% Configurando Motor A
velA = get(handles.slVelDisc, 'String');

mA = NXTMotor(MOTOR_A);

    mA.Power = (str2num(velA));
    mA.SendToNXT();

%% Set up ports
portaSensor = get(handles.slPortSensor, 'String');
if portaSensor == '1'
    portLuz = SENSOR_1;
elseif portaSensor == '2'
        portLuz = SENSOR_2;
elseif portaSensor == '3'
            portLuz = SENSOR_3;
elseif portaSensor == '4'
    portLuz = SENSOR_4;
end
%% Ativando Sensor de Cor como um Sensor de Luz
corSensor = get(handles.slCorSensor, 'Value');
if corSensor == 1
    OpenNXT2Color(portLuz, 'BLUE')
elseif corSensor == 2
        OpenNXT2Color(portLuz, 'RED')
elseif corSensor == 3
        OpenNXT2Color(portLuz, 'GREEN')
end
%% Parâmetros
count = 100;
x = 0 : 1 : count-1 ;
%% Array com os valores do Sensor e Gráfico
cla %limpando gráfico

for i = 1 : count

    pause(0.1)

    y(i) = GetNXT2Color(portLuz);

    hold on

    i = i + 1;
end
```

```
xlabel('Contador/Tempo')
ylabel('Valor Numérico (Cor)')

CorSinal = get(handles.slCorSinal, 'Value');

if CorSinal == 1
    plot(handles.slGraf,x,y,'b')
elseif CorSinal == 2
    plot(handles.slGraf,x,y,'m')
elseif CorSinal == 3
    plot(handles.slGraf,x,y,'k')
elseif CorSinal == 4
    plot(handles.slGraf,x,y,'r')
elseif CorSinal == 5
    plot(handles.slGraf,x,y,'g')
end

CloseSensor(portLuz);
mA.Stop('Off');
COM_CloseNXT(h);

guidata(hObject, handles);

% --- Executes on button press in slSair.
function slSair_Callback(hObject, eventdata, handles)
% hObject    handle to slSair (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)
s=questdlg('Deseja sair?','Confirmação','Sim','Não');

if s == 'Sim'

    clear all
    clc
    close all;

end
guidata(hObject, handles);


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)
%% Salvar Gráfico
    S = getframe(handles.slGraf);               %select axes in GUI
    figure();                           %new figure
    image(S.cdata);                             %show selected axes in
new figure
    saveas(gca, 'path', 'fig');                 %save figure
    %close(gcf)


% --- Executes on button press in slLimp.
function slLimp_Callback(hObject, eventdata, handles)
% hObject    handle to slLimp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)
```

```
cla %limpando gráfico
guidata(hObject, handles);
```

ANEXO C – Programa 6 (Gráfico do Sensor Ultrassônico Utilizando uma Interface Gráfica)

```
function varargout = Maintain_distance_grafico_GUI_SET(varargin)
% MAINTAIN_DISTANCE_GRAFICO_GUI_SET MATLAB® code for
Maintain_distance_grafico_GUI_SET.fig
%      MAINTAIN_DISTANCE_GRAFICO_GUI_SET, by itself, creates a new
MAINTAIN_DISTANCE_GRAFICO_GUI_SET or raises the existing
%      singleton*.
%
%      H = MAINTAIN_DISTANCE_GRAFICO_GUI_SET returns the handle to a new
MAINTAIN_DISTANCE_GRAFICO_GUI_SET or the handle to
%      the existing singleton*.
%
%
MAINTAIN_DISTANCE_GRAFICO_GUI_SET('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in MAINTAIN_DISTANCE_GRAFICO_GUI_SET.M with
the given input arguments.
%
%      MAINTAIN_DISTANCE_GRAFICO_GUI_SET('Property','Value',...) creates a
new MAINTAIN_DISTANCE_GRAFICO_GUI_SET or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before
Maintain_distance_grafico_GUI_SET_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to
Maintain_distance_grafico_GUI_SET_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help
Maintain_distance_grafico_GUI_SET

% Last Modified by GUIDE v2.5 31-Aug-2013 15:42:24

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@Maintain_distance_grafico_GUI_SET_OpeningFcn, ...
                   'gui_OutputFcn',
@Maintain_distance_grafico_GUI_SET_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
```

```
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Maintain_distance_grafico_GUI_SET is made
visible.
function Maintain_distance_grafico_GUI_SET_OpeningFcn(hObject, eventdata,
handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Maintain_distance_grafico_GUI_SET
(see VARARGIN)

% Choose default command line output for Maintain_distance_grafico_GUI_SET
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Maintain_distance_grafico_GUI_SET wait for user response
(see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Maintain_distance_grafico_GUI_SET_OutputFcn(hObject,
eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in mdCom.
function mdCom_Callback(hObject, eventdata, handles)
% hObject    handle to mdCom (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)
%% Iniciando comunicação com o LEGO
%Check toolbox installation: verify that the RWTH - Mindstorms NXT toolbox
is installed.
if verLessThan('RWTHMindstormsNXT', '2.00');
    error(strcat('This program requires the RWTH - Mindstorms NXT Toolbox '
...
        ,'version 3.00 or greater. Go to http://www.mindstorms.rwth-
aachen.de ' ...
        ,'and follow the installation instructions!'));
end%if

% Clean up: Close previous handles (if existing)
%close all
clear all
clc
format compact
```

```
% Iniciando comunicação com o LEGO
h = COM_OpenNXT();
COM_SetDefaultNXT( h );
NXT_PlayTone( 440, 100 )


function mdPorUS_Callback(hObject, eventdata, handles)
% hObject    handle to mdPorUS (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)
portSensor = str2num(get(hObject,'String'));
if (isempty(portSensor))
    set(hObject, 'String','0')
    %msgbox('Digite um valor numérico…', 'Atenção');
    errordlg('Por favor, digite valor numérico.', 'Erro');
else
    if portSensor < 1 || portSensor > 4
        set(hObject, 'String','0')
        %msgbox('Digite um valor entre 0 e 100…', 'Atenção');
        errordlg('Por favor, digite um número válido.', 'Erro');
    else
       guidata(hObject, handles);
    end
end
guidata(hObject, handles);

% Hints: get(hObject,'String') returns contents of mdPorUS as text
%        str2double(get(hObject,'String')) returns contents of mdPorUS as a
double


% --- Executes during object creation, after setting all properties.
function mdPorUS_CreateFcn(hObject, eventdata, handles)
% hObject    handle to mdPorUS (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function mdDist_Callback(hObject, eventdata, handles)
% hObject    handle to mdDist (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)
dist = str2num(get(hObject,'String'));
if (isempty(dist))
    set(hObject, 'String','15')
    %msgbox('Digite um valor numérico…', 'Atenção');
    errordlg('Por favor, digite valor numérico.', 'Erro');
else
    if dist < 15 || dist > 50
        set(hObject, 'String','15')
        %msgbox('Digite um valor entre 0 e 100…', 'Atenção');
        errordlg('Por favor, digite um número válido.', 'Erro');
    else
```

```
        guidata(hObject, handles);
    end
end
guidata(hObject, handles);

% Hints: get(hObject,'String') returns contents of mdDist as text
%        str2double(get(hObject,'String')) returns contents of mdDist as a
double


% --- Executes during object creation, after setting all properties.
function mdDist_CreateFcn(hObject, eventdata, handles)
% hObject    handle to mdDist (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in mdCorSinal.
function mdCorSinal_Callback(hObject, eventdata, handles)
% hObject    handle to mdCorSinal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns mdCorSinal
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
mdCorSinal


% --- Executes during object creation, after setting all properties.
function mdCorSinal_CreateFcn(hObject, eventdata, handles)
% hObject    handle to mdCorSinal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in mdIniciar.
function mdIniciar_Callback(hObject, eventdata, handles)
% hObject    handle to mdIniciar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)
%% Iniciando comunicação com o LEGO
h = COM_OpenNXT();
COM_SetDefaultNXT( h );

%% Configurando Motores AB
```

```
mBC = NXTMotor( [MOTOR_B; MOTOR_C] );
mBC.SmoothStart = true;

%% Set up ports
portUS = get(handles.mdPorUS, 'String');
if portUS == '1'
    portUS2 = SENSOR_1;
elseif portUS == '2'
        portUS2 = SENSOR_2;
elseif portUS == '3'
            portUS2 = SENSOR_3;
elseif portUS == '4'
    portUS2 = SENSOR_4;
end
%% Ativando Sensor US
OpenUltrasonic(portUS2, 'snapshot')
%% Distância a ser mantida
n = get(handles.mdDist, 'String');
target = (str2num(n));
%% Parâmetros
count = 100;
x = 0 : 1 : count-1 ;
%% Array com so valores do Sensor e Gráfico
cla %limpando gráfico
legend(handles.mdGraf,'off')

for i = 1 : count

    USMakeSnapshot(portUS2)
    pause(0.05)

    d = GetUltrasonic(portUS2);%USGetSnapshotResults( portUS2 ); %
Distância para acionamento do motor

    y(i) = d;%GetUltrasonic(portUS2);

    hold on

    i = i + 1;

% Movimento do Robô
        mBC.Power = floor( (d - target) * 10);
        mBC.SendToNXT();

        pause( 0.1 )

end
xlabel('Contador/Tempo')
ylabel('Distância (cm)')

CorSinal = get(handles.mdCorSinal, 'Value');

if CorSinal == 1
    plot(handles.mdGraf,x,y,'b')
elseif CorSinal == 2
    plot(handles.mdGraf,x,y,'c')
elseif CorSinal == 3
    plot(handles.mdGraf,x,y,'m')
```

```
elseif CorSinal == 4
    plot(handles.mdGraf,x,y,'r')
elseif CorSinal == 5
    plot(handles.mdGraf,x,y,'g')
end
%hold on
plot(handles.mdGraf,x,target,'k')
legend('Sensor US', 'Set Point', 'Location','NW');

%% Clean up
mBC.Stop('Off')
CloseSensor(portUS2)
COM_CloseNXT(h);
%end
guidata(hObject, handles);


% --- Executes on button press in mdSair.
function mdSair_Callback(hObject, eventdata, handles)
% hObject    handle to mdSair (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)
s=questdlg('Deseja sair?','Confirmação','Sim','Não');

if s == 'Sim'

    clear all
    clc
    close all;

end
guidata(hObject, handles);


% --- Executes on button press in mdSave.
function mdSave_Callback(hObject, eventdata, handles)
% hObject    handle to mdSave (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)
%% Salvar Gráfico
    S = getframe(handles.mdGraf);               %select axes in GUI
    figure();                           %new figure
    image(S.cdata);                                 %show selected axes in
new figure
    saveas(gca, 'path', 'fig');                     %save figure
    %close(gcf)



% --- Executes on button press in mdLimp.
function mdLimp_Callback(hObject, eventdata, handles)
% hObject    handle to mdLimp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB®
% handles    structure with handles and user data (see GUIDATA)
cla %limpando gráfico
legend(handles.mdGraf,'off')
guidata(hObject, handles);
```